1.1 Institute Profile

Institute of Management and Career Courses (IMCC) is a premier Management Institute, established in 1983 by Maharashtra Education Society (MES) for providing quality education and technical expertise at the Post Graduation Level in the Fields of Computers and Management. The Institute is recognized by SPPU under Section 46 of Pune University Act, 1974 and Section 85 of Maharashtra University Act, 1994 and Approved by AICTE New Delhi to conduct MCA and MBA programs. The Institute is located at 131, Mayur Colony, Kothrud, Pune-411038 having 30,000sq.ft. built area & totally independent campus.

IMCC is recognized as a Ph.D. Research Center under the Faculty of Management, SPPU. IMCC has 38 years standing & it is well-known for its conducive educational atmosphere. IMCC focuses on the all-round development of its students. Thus, apart from excellence in academics, students develop their inner potential by way of active participation in cocurricular & extra-curricular activities.

Guest Lectures, Seminars, Workshops, Industrial Visits& Placements. The main motto of the Institute is to instill the concepts of total personality development in the students. The emphasis is laid on

‘Teacher Disciple Relationship’ in place of ‘Boss Subordinate’ relationship at their assignment.

The preamble of IMCC “FACTA-NON-VERBA" lucidly means that the Institute produces the new breed of professionals, who’s deeds will speak and there could be no requirement of pomposity. The conducive milieu of the Institute molds the budding managers to reveal in managing flexibility, integration, change and transformation. These ‘would be’ professionals are channelized in such a way to ‘orchestrate’ and deploy business and technological management skills in a synergistic manner to grab the tangible success. The faculty members put their relentless efforts in educating the students to synthesize business management acumen and technology insights in a creative manner.

1.2 Abstract

Fixitwala: A Home Services App

Now a days for any services like Cleaning, Plumbing, Electrical, Carpentry, Home Paint and Machine Repairing, if any customer wants to use this type of services, then they can go through a personal meeting or mobile call. It is difficult for customer to find any service in emergency at any time and place.

So, with this project I am going to develop Android and iOS app which will help customers to find out solution for any problems related to Cleaning, Plumbing, Electrical and Carpentry. This application will provide a platform for the above house hold services at any time and place. The project will also provide the facilities like security, online payment, map navigation and also Notification.

1.3 Existing System and Need for System

I have observed limitations in existing system:

- Existing system is offline.
- Difficult to manage records.
- No time limit for service to be provided.
- No guarantied service.
- Difficult to find service provider.
- 24 hours service is not available.
- No security.

So, my purpose is to overcome these limitations with following features:

- Household services easily available.
- To provide house hold services any time.
- Easy online payment.
- Saving of time.
- Make available household services through mobile application.

1.4 Scope of System

The scope of our project is to designing a complete environment to provide a safe and user friendly environment for online service booking. The main aim of the project is to provide an easy to use application for services provided for customers.

We often get frustrated while taking the appointment of service provider because there many problems occur, like the service provider is busy at somewhere else or is not receiving our call or the cost is very high according to problem. So in this project we will remove this headache.

1.5 Operating Environment - Hardware and Software

Hardware:

- Processor: Intel Core i3 and above
- RAM: 4 GB or more
- Hard disk: 250 GB and more
- Android Phone:
    - RAM: 4GB and more
    - Storage: 16GB and more
- iPhone:
    - RAM: 2GB and more
    - Storage: 8GB and more

Software:

- Editor: Project IDX
- Operating System:
    - Windows 7 and above

    - Android Marshmallow (6.0) and above
    - iOS 8 and above

Web Browser:

- Microsoft Edge, Google Chrome, Mozilla Firefox, Brave browser

Documentation:

- Microsoft Word 2010 and above

1.6 Brief Description of Technology Used:

- The Frontend: Flutter

  Flutter is a UI toolkit from Google used for building natively compiled applications for mobile, web, and desktop from a single codebase. It allows for fast development and expressive UIs through its reactive framework.

- Backend: Dart

  Dart is a programming language developed by Google, primarily used with Flutter for backend development. It's known for its fast performance and scalability, and it's particularly well-suited for building server-side applications.

- State management: setState

  "setState" is a method used in Flutter for managing the state of widgets. It triggers a rebuild of the widget tree when the state changes, updating the UI accordingly. While it's the simplest form of state management in Flutter, it's suitable for small to medium-sized applications.

- Database: SQLite

  SQLite is a lightweight, serverless, self-contained SQL database engine. It's widely used in mobile and desktop applications for local storage. In Flutter applications, SQLite can be integrated using plugins to provide data persistence and offline capabilities.

2.1 Study of Similar Systems

By studying these similar systems and related research papers, we can gain insights into best practices, user behaviours, and technological innovations in the home services sector, thereby informing the design and implementation of the platform.

1. Handy: Handy is a popular platform connecting users with various home service professionals, including cleaners, handymen, plumbers, and electricians.
2. TaskRabbit: TaskRabbit is a service marketplace where users can find help for various tasks around the home, such as furniture assembly, moving assistance, and minor repairs. It offers vetted Taskers and flexible scheduling.
3. Thumbtack: Thumbtack connects users with local professionals for a wide range of home services, including home improvement, event planning, and personal wellness. It emphasizes personalized quotes and customer reviews.
4. HomeAdvisor: HomeAdvisor is a platform that helps users find and hire contractors for home renovation and repair projects. It offers verified reviews, cost estimates, and project planning tools.
5. Pro.com: Pro.com is a platform that simplifies the process of finding and hiring home service professionals for projects like remodelling, landscaping, and maintenance.

2.2 Feasibility Study:

Feasibility study is carried out when there is a complex problem or opportunity. It is considered as the primary investigation which emphasizes on "Look before You Loop" approach to any project .A Feasibility study is undertaken to determine the possibility of either improving the existing system or developing a completely new system.

I am going to develop a new system which is feasible as our application is very user friendly and easy to understand.

2.2.1 Technical Feasibility:

In this type of study, the current technology in used in an organization is checked such as the existing software, hardware, and personnel staff to determine whether it will work for the proposed system or completely new ones are to be used. The technology that was important in developing a new system such as Development tools, back-end database system was available from within the organization. The proposed system is capable of adding, changing, enhancing functionality, features etc. The proposed system is capable of handling large storage of data. The back-end and front-end technology has greater important for providing an accurate, error-free, frequencies of data to be used.

My project is technically feasible in terms of current technology. It will provide latest platform like android and iOS technology.

2.2.2 Financial Feasibility:

For proving that system developed is economical, the economic feasibility study takes place to check the cost of developing a system against the benefits that it provides. If the cost is less and benefits are more then we can define our system to be economically developed. User saves time in searching for a particular product to be purchased by simply few clicks. The registration process is speedier than the registered manually. The saving of papers as all data are stored computerized. The record is of free of human errors as there is less chance of mistakes. The above benefits are in terms of saving time, minimize errors and provide efficiency in work done.

In terms of economic feasibility this application is very reasonable in cost. So, application is economically feasible.

2.2.3 Operational Feasibility:

The operational feasibility is concerned with the operability of the system after it has been installed. That is, some programmer may not like changes in their routine method of work or has fear that they will lose their peer group. The following areas will have the operational feasibility in the proposed project

- The organization has approved this system as their working system.
- The User of the system has accepted the proposed system as their new working system and realized the benefits of it.
- The system will work in a proper way after it has been installed and the installation process is easy to use.

2.3 Objectives of Proposed System:

The objectives of the proposed "Fixitwala: A home services app " are as follows:

- Efficient Service Booking System
- Real-time Availability Check
- Transparent Service Information
- Timely and Reliable Services
- Customer Convenience
- User-Friendly Interface
- Use of Latest Technologies

2.4 Users of the System (app):

1. Admin

Administrator has maximum privileges to access the system. He maintains user login details, can assign access rights to a user, can manipulate data and do all the transactions. Administrator is the super-user of the system.

Following are the tasks that Admin performs:

- Verify service provider and customer.
- Manage all categories of service.
- Can send notification to the customer and service provider.
- View services and feedback

2. Service Provider

In this application, the initial step involves registration, followed by a login process. Upon completion, the service provider gains access to a dashboard displaying the services ordered by users. Subsequently, the service provider sends a positive acknowledgement to the user.

Upon confirmation, the service provider obtains a unique BookingId, which corresponds to the customer's BookingId. Once at the customer's location, the service provider verifies the BookingId, ensuring a secure match with the customer's code, before proceeding with the designated service.

3. Customer

In this application, the customer begins by completing the registration process, followed by logging into the system. Once logged in, the user can search for a specific service, leading to the retrieval of a list of available services via our Android application. After choosing a desired service, the user initiates a service request.

Upon making the request, the customer receives an acknowledgement in the form of a reply. Additionally, a unique BookingId is generated for each user as part of the confirmation process. This BookingId serves as an exclusive identifier for the user and is pivotal for subsequent interactions within the application.

3.1 System Requirements (Functional and Non-Functional requirements)

System Requirements for the "Fixitwala" app platform can be categorized into functional and non-functional requirements:

Functional Requirements:

1. User Registration and Authentication:

- Users should be able to register for an account on the platform using email and password.

- Authentication mechanisms should be in place to verify user identity and ensure security.

2. Service Listings:

- Service Providers should be able to create, edit, and manage service listings, including detailed descriptions, photographs, amenities, and pricing information.

- Customers should be able to browse, search, and filter services based on various criteria such as location, budget, and category.

3. Communication Tools:

- Integrated chat system allowing direct communication between brokers and buyers.
- Direct phone calls to provide assistance and answer user queries.

4. Service Viewing and Booking:

- Customers should be able to schedule service viewings directly through the platform.
- Service Provider should be able to confirm, reschedule, or cancel service appointments.

5. User Profiles and Preferences:

- User profiles with personalized settings, saved searches, favorite listings, and order history.

Non-Functional Requirements:

1. Performance:

- Fast response times and minimal latency for loading and executing user actions.

- Scalability to handle increasing user traffic and data volume over time.

2. Security:

- Robust security measures to protect user data, transactions, and sensitive information.

- Encryption of data transmissions and storage to prevent unauthorized access.

3. Reliability:

- High availability and uptime of the platform to ensure uninterrupted access for users.

- Backup and recovery mechanisms to prevent data loss in case of system failures or disasters.

4. Usability:

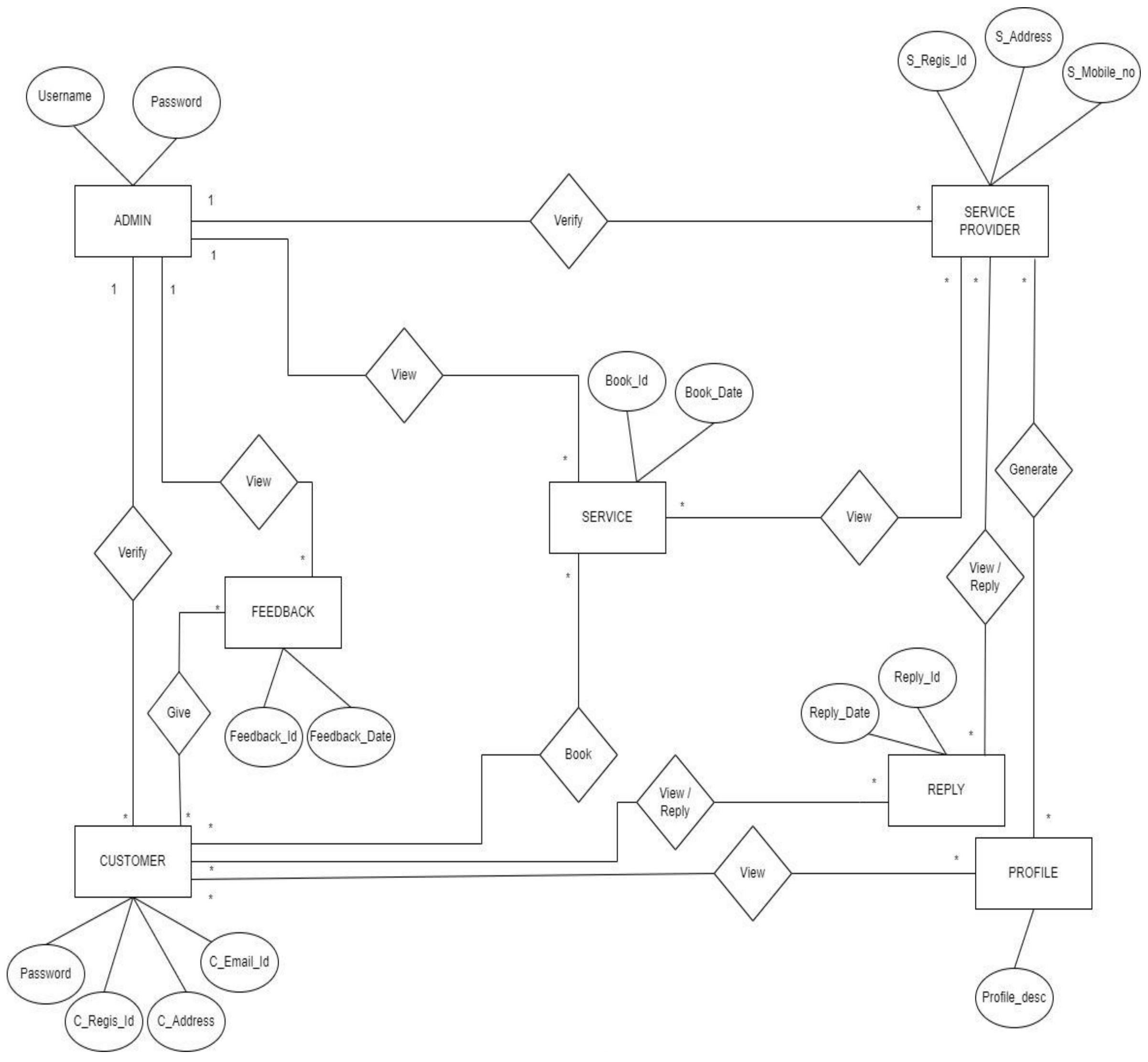
- Intuitive and user-friendly interface design to enhance user experience and usability.

- Accessibility features to ensure inclusivity for users with disabilities.

5. Compatibility:

- Compatibility with various mobile operating systems

3.2 Entity Relationship Diagram (ERD)



3.3 Table Structure

1. **ADMIN LOGIN TABLE:**

SNO	COLUMN	DATATYPE	CONSTRAINTS	DESCRIPTION
1	ADMIN_ID	VARCHAR (50)	PRIMARY KEY	Unique id for Admin
2	EMAIL	VARCHAR (50)		Admin email id
3	PASSWORD	VARCHAR (10)		Admin password

2. **CUSTOMER_REGISTRATION TABLE:**

SR NO	COLUMN	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	C_ID	INT (10)	PRIMARY KEY	customer id
2	C_NAME	VARCHAR (50)		customer name
3	PROFILE PIC	VARCHAR (50)		customer profile pic
4	ADDRESS	VARCHAR (255)		customer address
5	MOBILE_NO	VARCHAR (10)		customer phone no.
6	EMAIL_ID	VARCHAR (50)		customer email id
7	PASSWORD	VARCHAR (30)		login password
8	VERIFICATION _STATUS	INT (1)		customer verification status (verified/not)

3. SERVICE_PROVIDER_REGISTRATION TABLE:

SNO	COLUMN	DATATYPE	CONSTRAINTS	DESCRIPTION
1	SP_ID	VARCHAR (10)	PRIMARY KEY	SP unique id
2	SP NAME	VARCHAR (50)		SP name
3	PROFILE PIC	VARCHAR (50)		SP profile pic
4	ADDRESS	VARCHAR (255)		SP address
5	MOBILE NO	VARCHAR (10)		SP mobile no
6	EMAIL ID	VARCHAR (50)		SP email id
7	PASSWORD	VARCHAR (30)		SP loginpassword
8	CATEGORY	VARCHAR (10)		service category
9	DESCRIPTION	VARCHAR (255)		details of service
10	DURATION	VARCHAR (10)		time of available
11	VERIFICATION_ STATUS	INT (1)		verification status (verified/not)

4. **ORDER_SERVICE TABLE:**

SR NO	COLUMN	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	ORDER_ID	INT (10)	PRIMARY KEY	unique id for order
2	C_ID	INT (10)	FOREIGN KEY	customer id
3	SP_ID	INT (10)	FOREIGN KEY	SP id from SP table
4	SERVICE_ID	INT (10)	FOREIGN KEY	unique id of service
5	BOOK_DATE	DATE		date of booking
6	STATUS	INT (1)		service given or not

5. **SERVICE TABLE:**

SR NO	COLUMN	DATA TYPE	CONSTRAINTS	DESCRIPTION
1	SERVICE_ID	INT (10)	PRIMARY KEY	unique id for service
2	SERVICE NAME	VARCHAR (50)		service name
3	SP_ID	VARCHAR (10)	FOREIGNKEY	SP id from SP table
4	CATEGORY	VARCHAR (50)	FOREIGN KEY	service category
5	SERVICE_IMAGE	VARCHAR (50)		Image of Service
6	FIXED RATE	VARCHAR (3)		Fixed amount i.e. 300

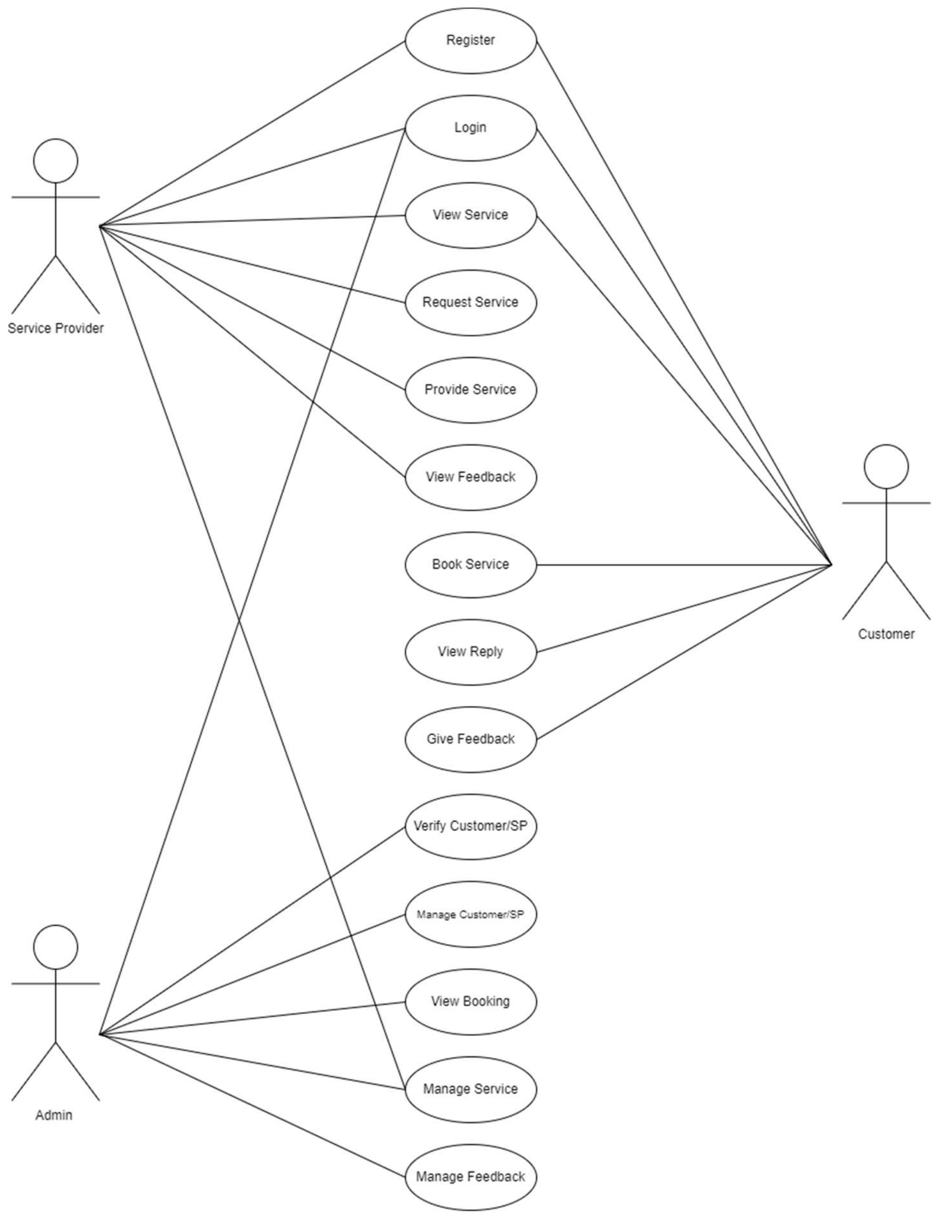
6. **REPLY_DETAIL TABLE:**

SR NO	**COLUMN**	**DATA TYPE**	**CONSTRAINTS**	**DESCRIPTION**
1	REPLY_ID	INT (10)	PRIMARY KEY	unique id of each reply
2	REPLY_DATE	DATE		date of reply
3	SP_ID	INT (10)	FOREIGN KEY	SP id from SP table
4	C_ID	INT (10)	FOREIGN KEY	Customer id
5	REPLY_MSG	VARCHAR (255)		message of reply
6	SENDER_TYPE	VARCHAR (50)		customer or SP

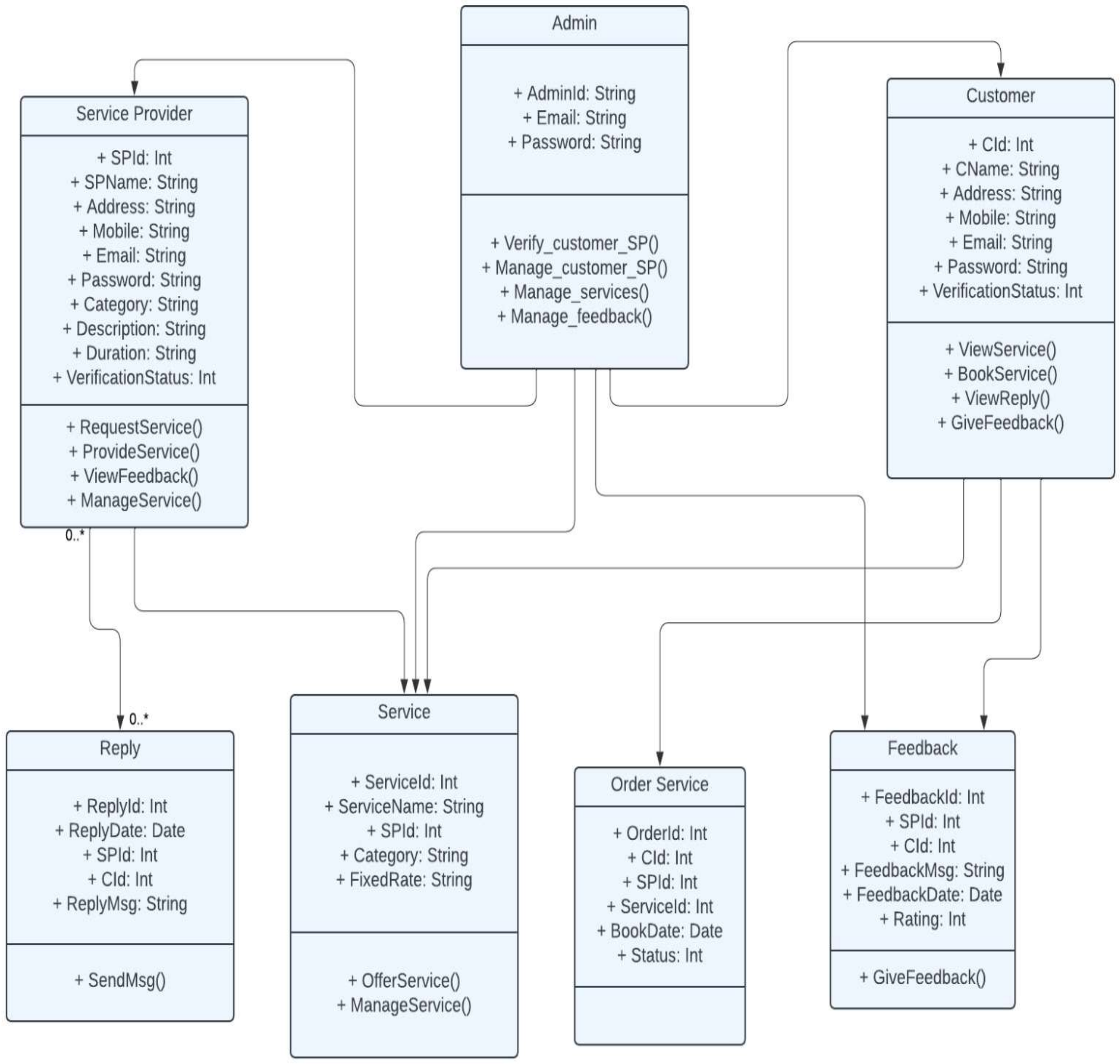
7. **FEEDBACK TABLE:**

SR NO	**COLUMN**	**DATA TYPE**	**CONSTRAINTS**	**DESCRIPTION**
1	FEEDBACK_ID	INT (10)	PRIMARY KEY	unique id
2	SP_ID	INT (10)	FOREIGN KEY	SP id from SP table
3	C_ID	INT (10)	FOREIGN KEY	customer id
4	FEEDBACK MESSAGE	VARCHAR (255)		message sent as feedback
5	FEEDBACK_DATE	DATE		date of message sent
6	RATING	INT (1)		rating 1-5

3.4 Use Case Diagram

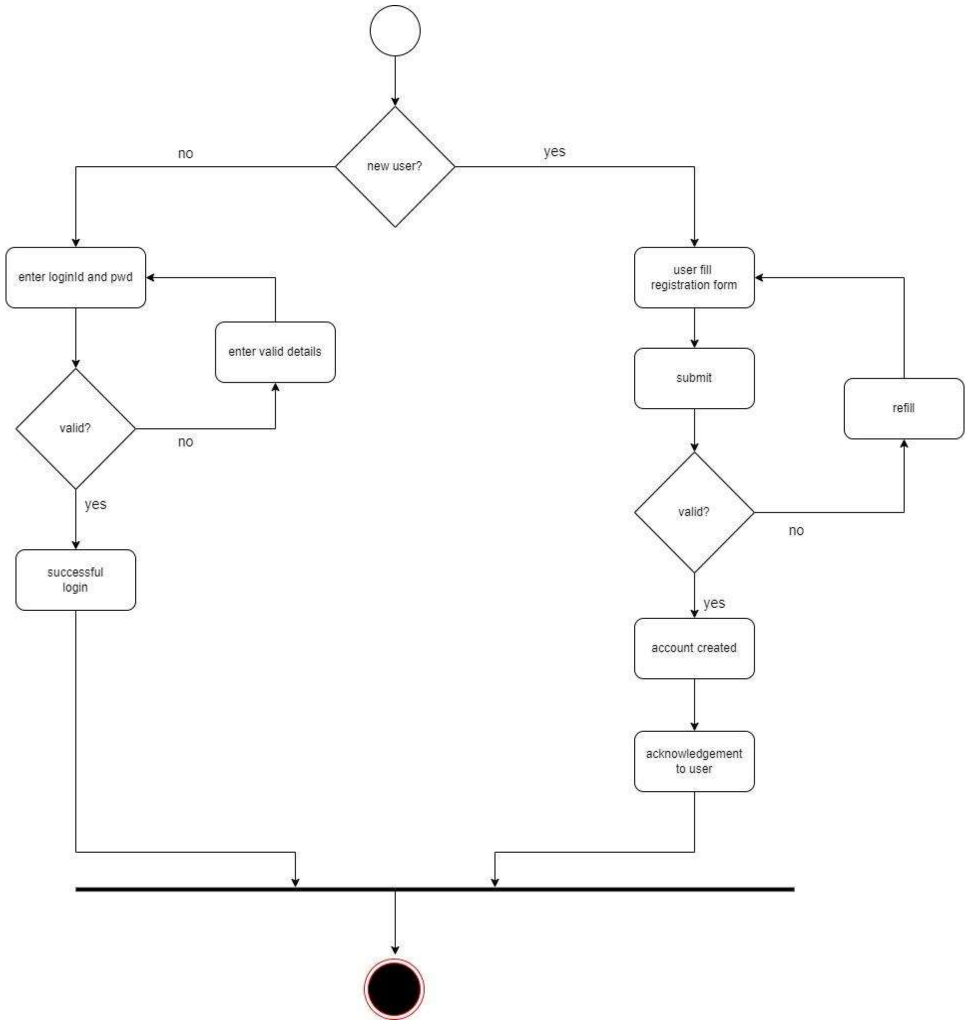


3.5 Class Diagram

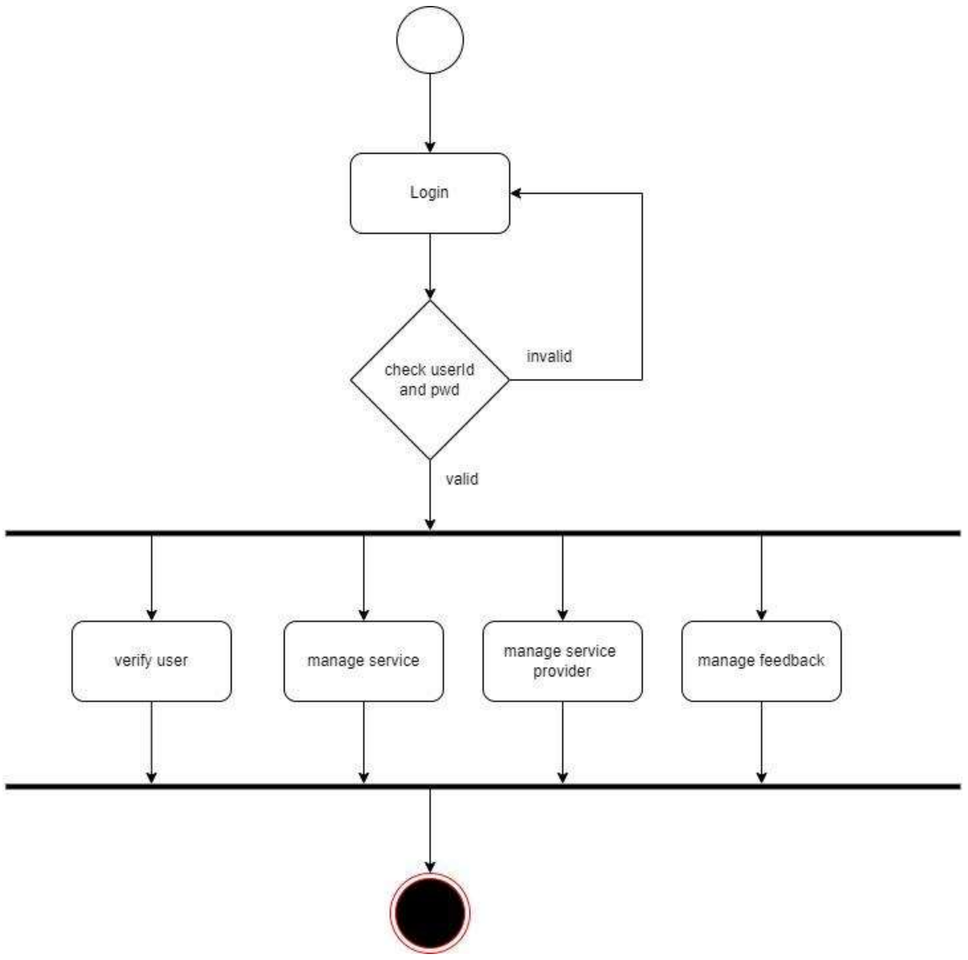


3.6 Activity Diagram

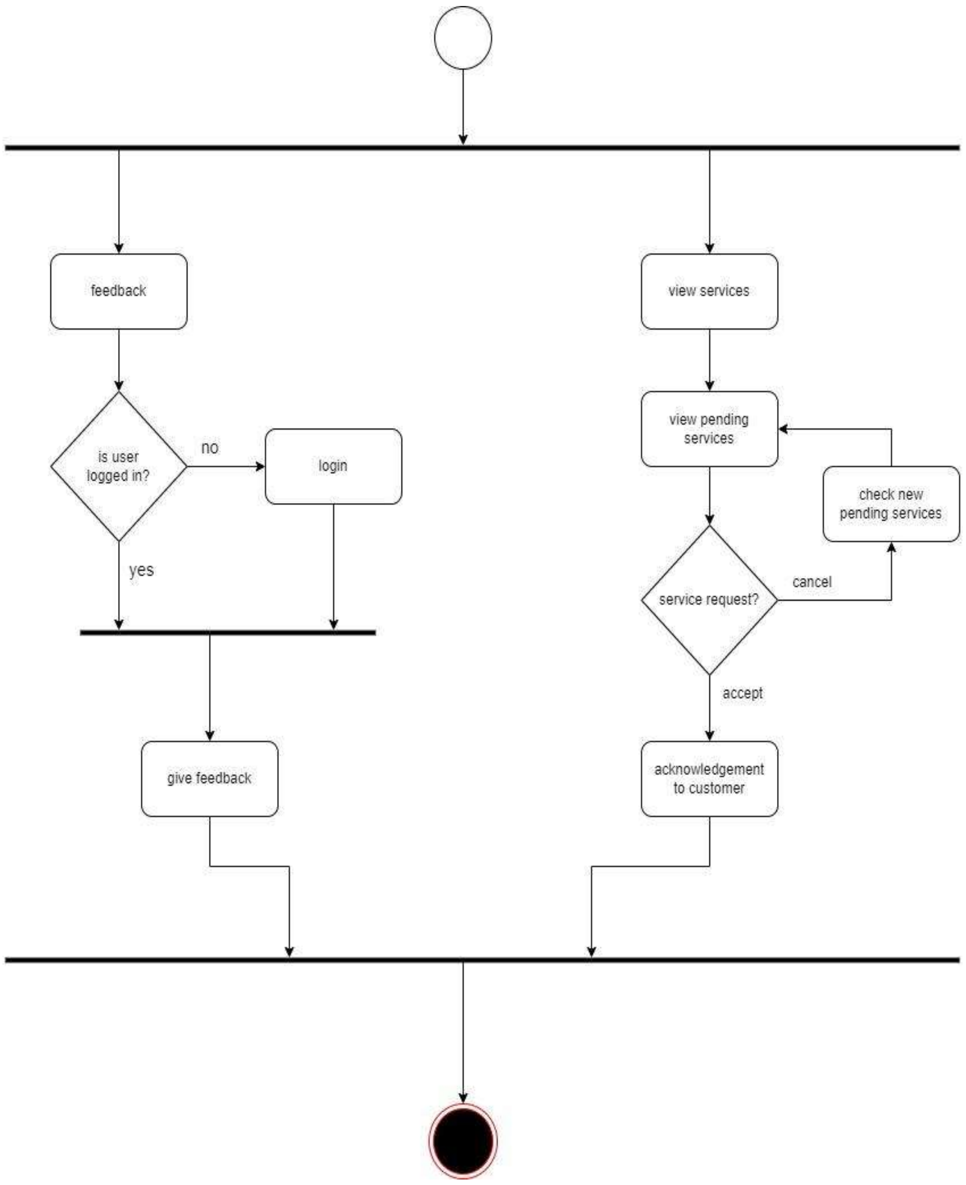
Login & Registration



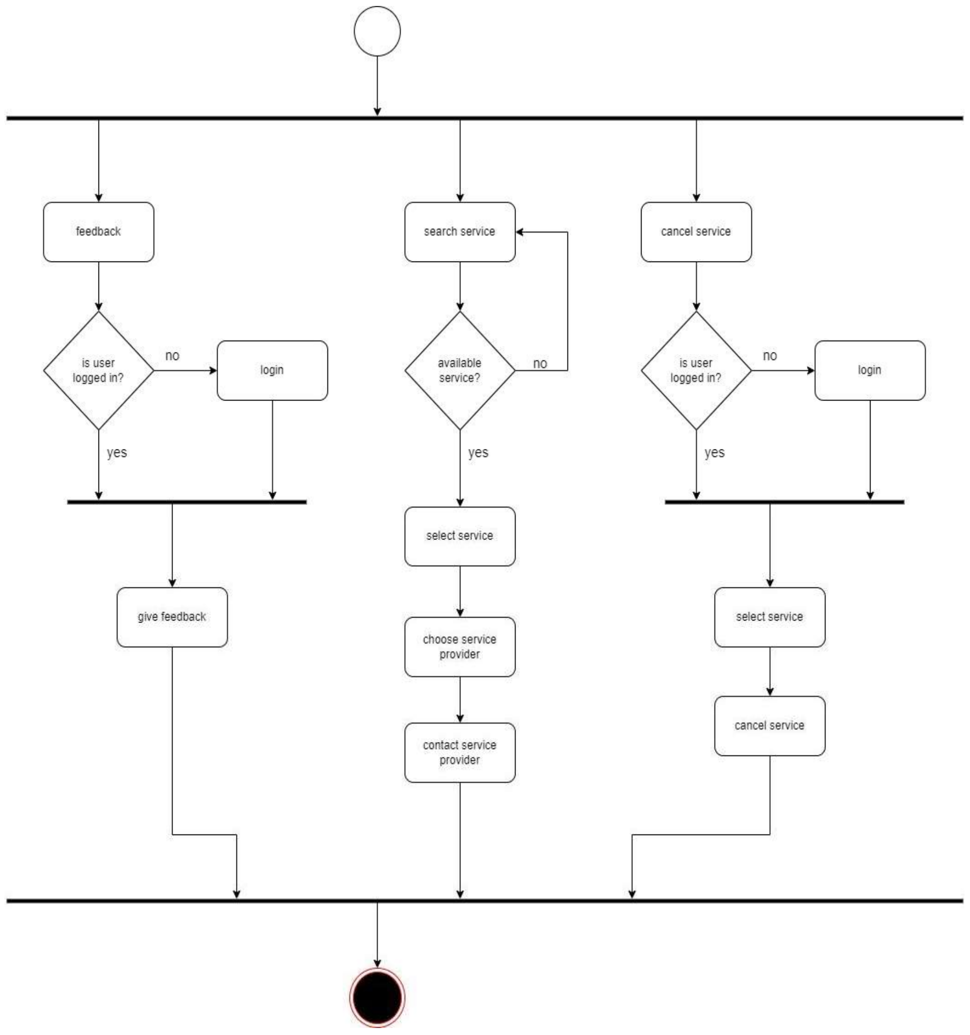
Admin


Login
check userId and pwd
invalid
valid
verify user
manage service
manage service provider
manage feedback

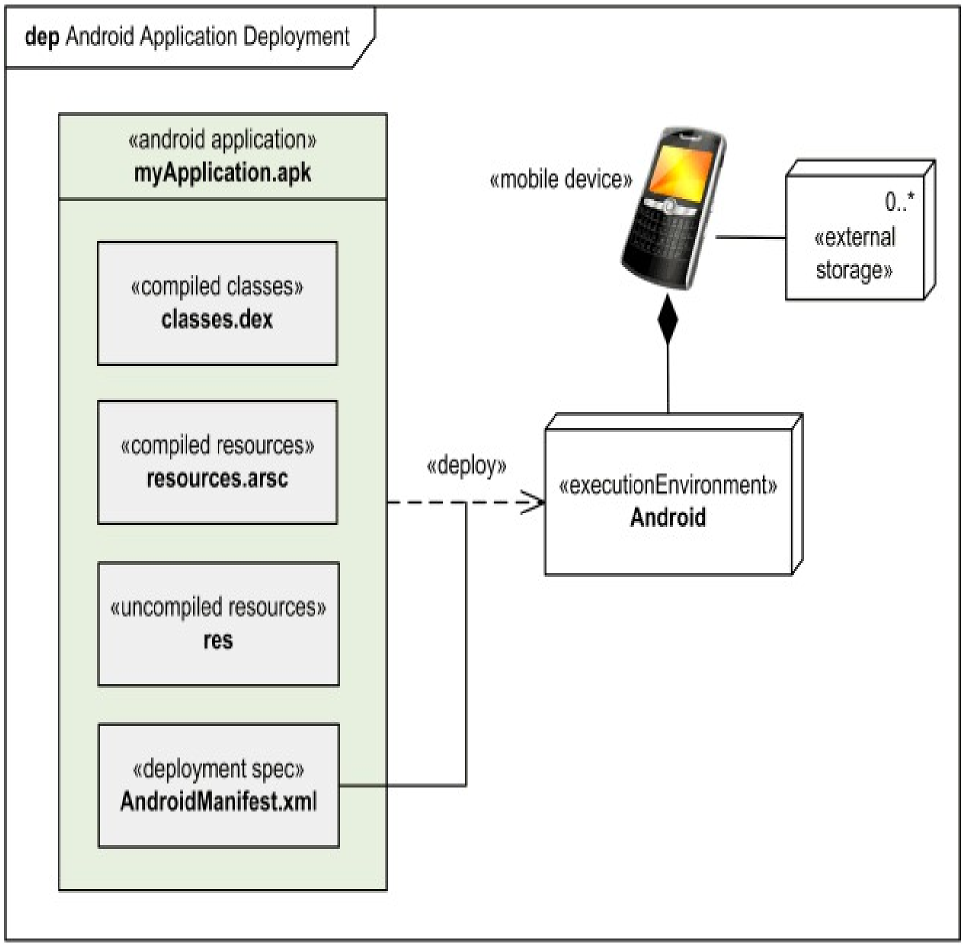
Service Provider



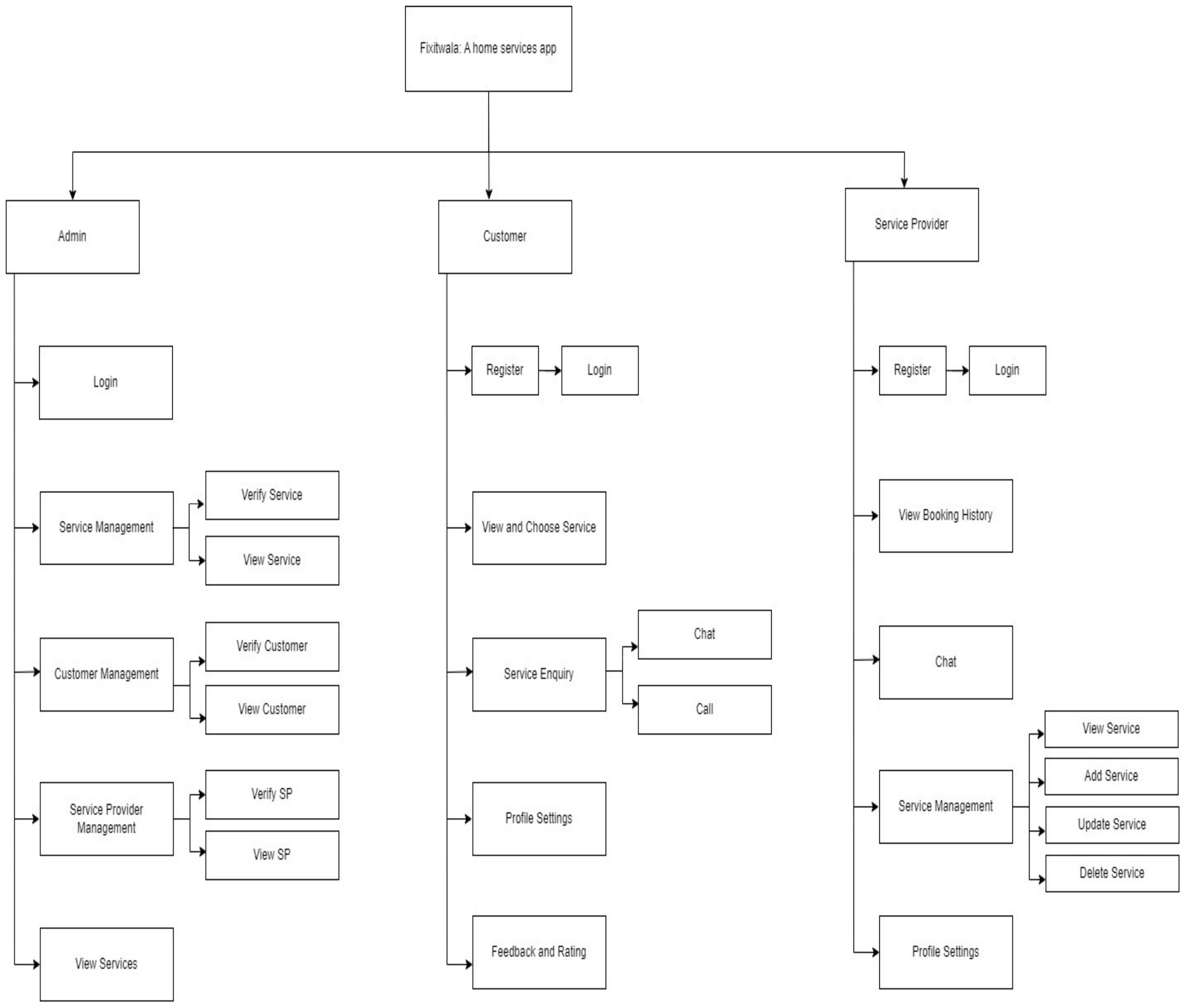
Customer


feedback
search service
cancel service
is user logged in?
no
login
available service?
no
is user logged in?
no
login
yes
yes
yes
select service
give feedback
select service
choose service provider
cancel service
contact service provider

3.7 Deployment Diagram

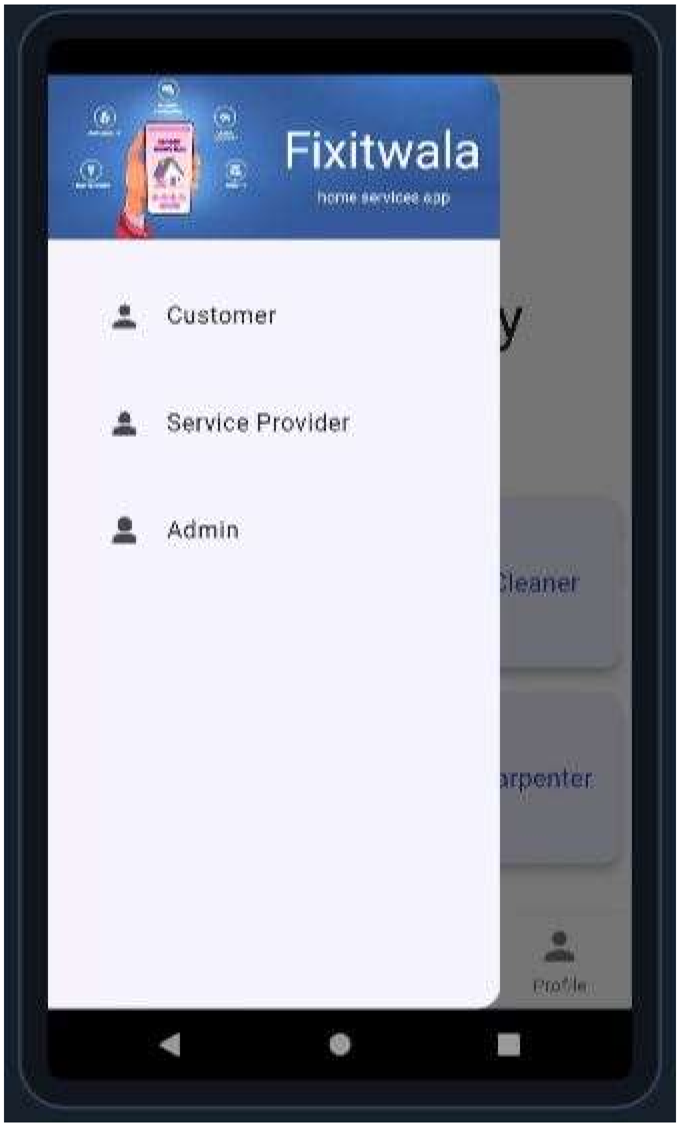


3.8 Module Hierarchy Diagram

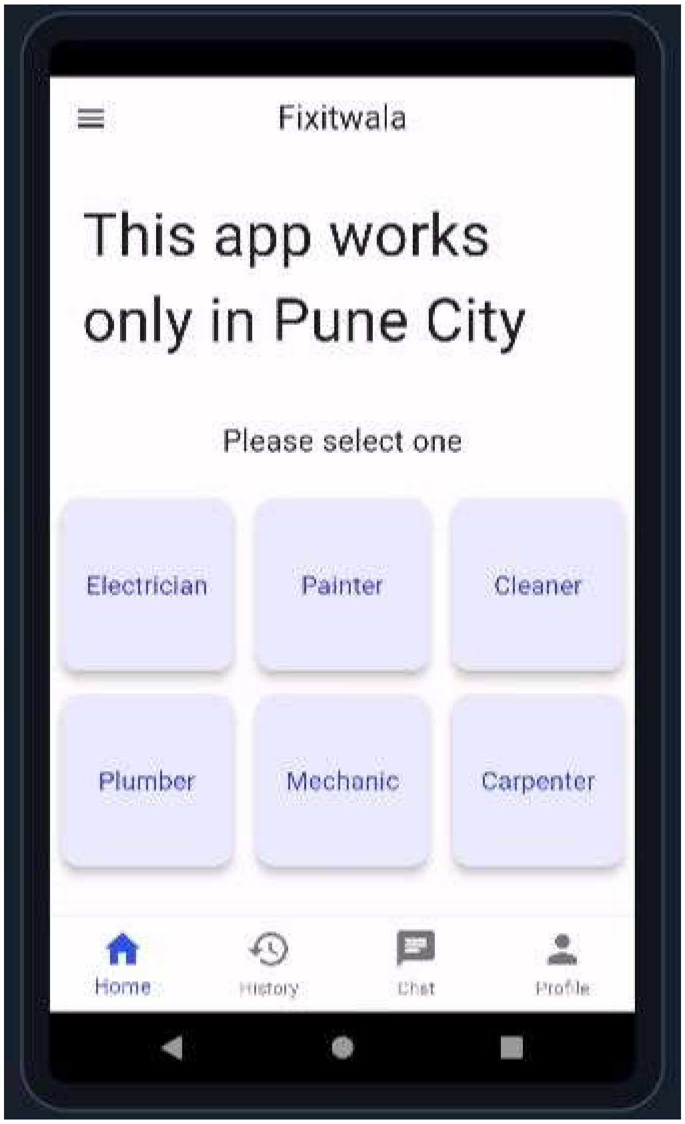


3.9 Sample Input and Output Screens

Home Page Drawer



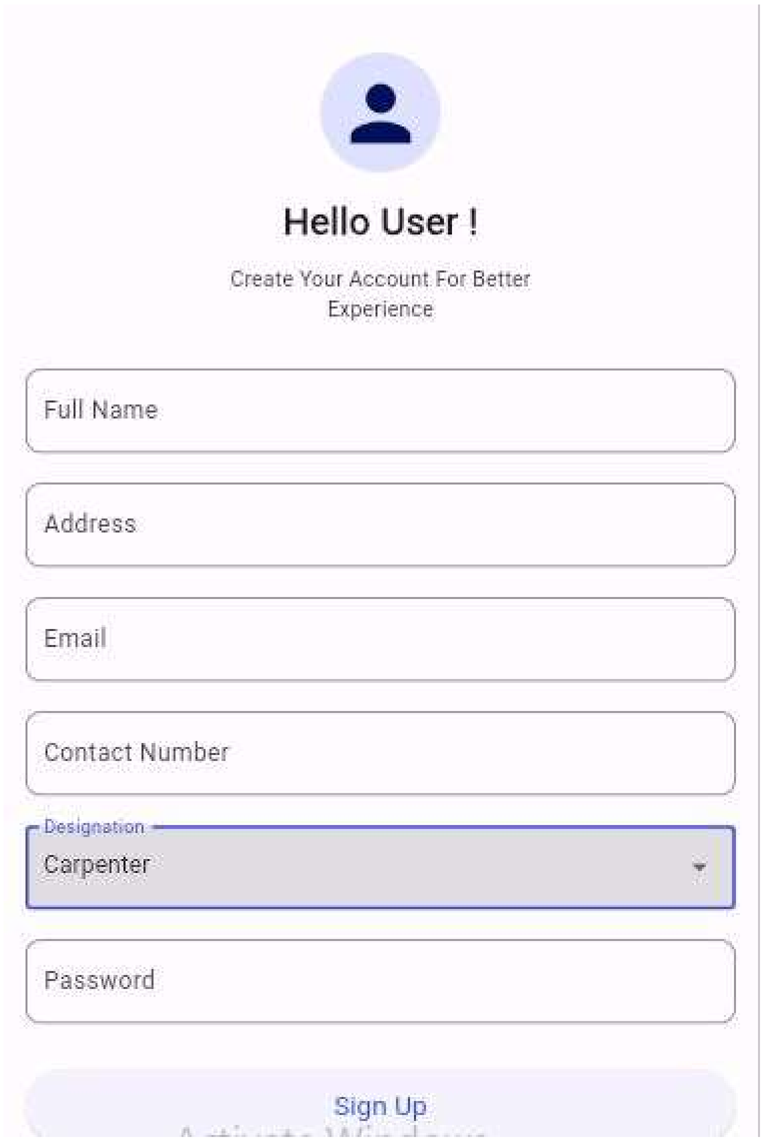
Home Page



This app works only in Pune City

Please select one

Service Provider Registration Page



Hello User !

Create Your Account For Better Experience

Full Name

Address

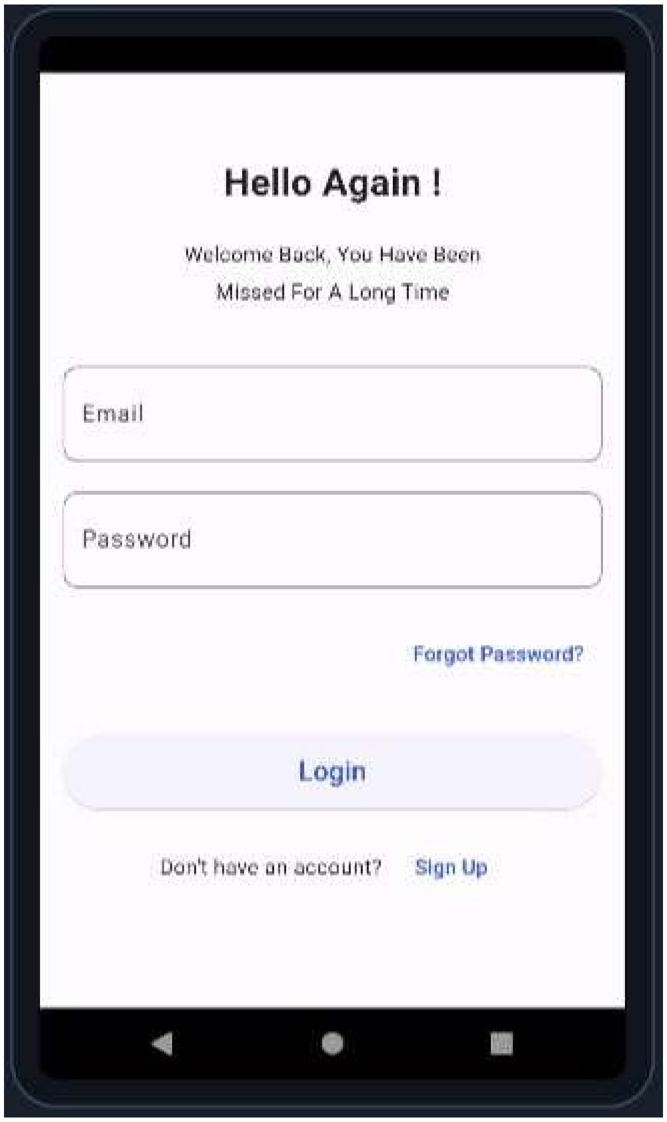
Email

Contact Number

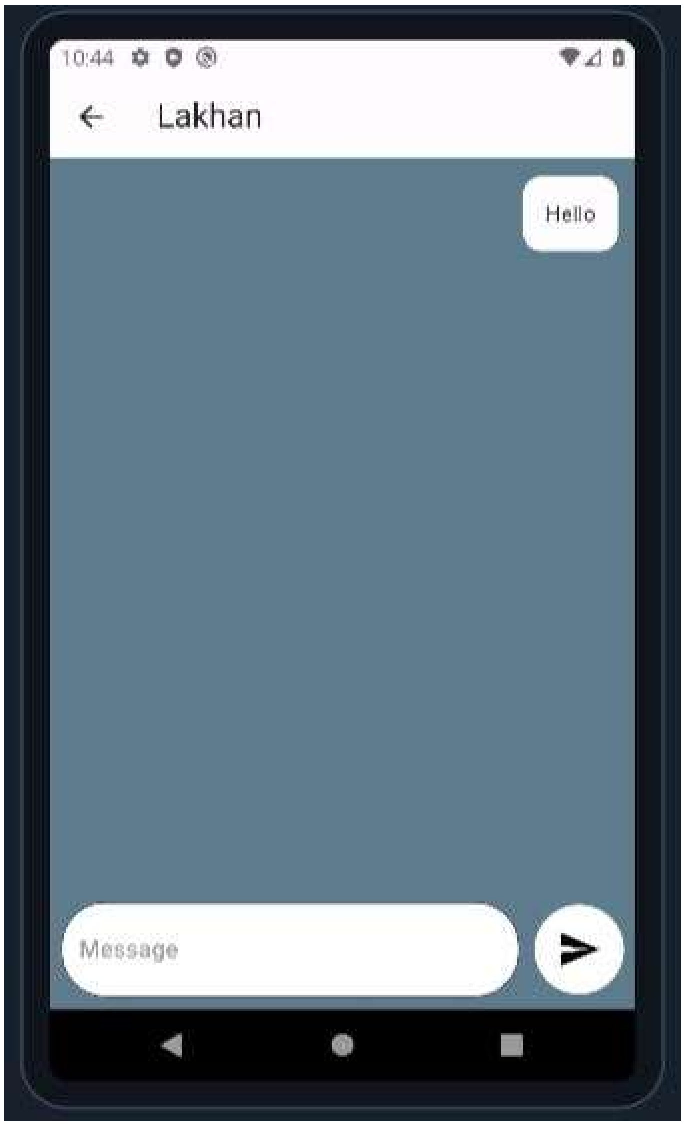
Designation: Carpenter

Password

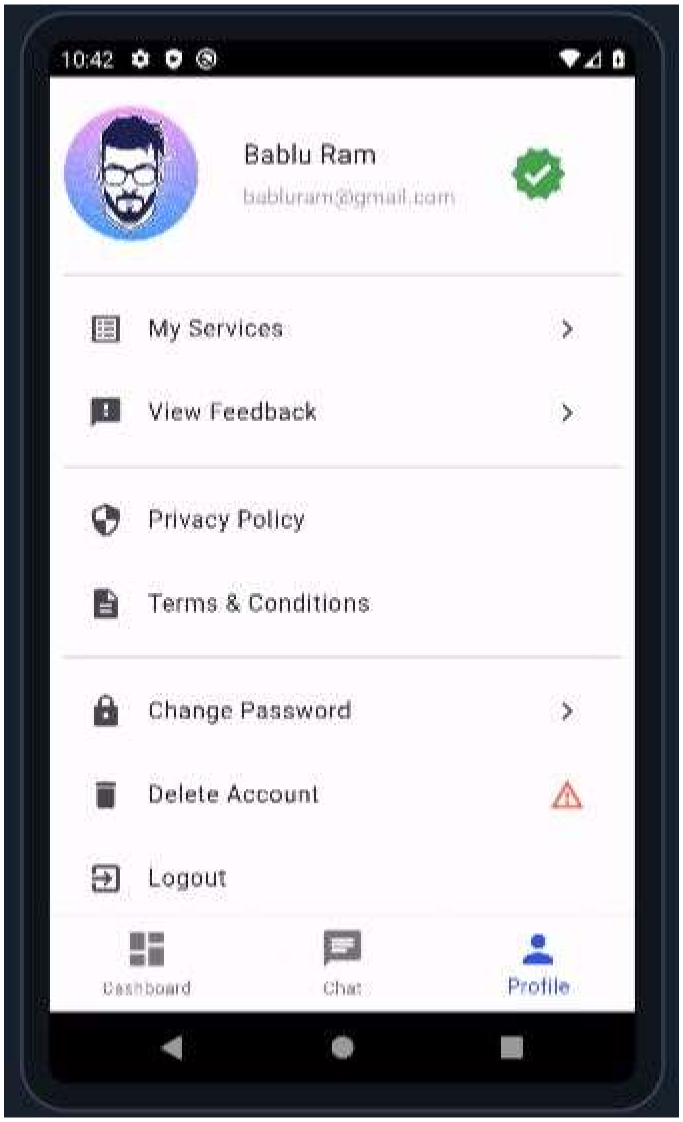
Service Provider Login Page



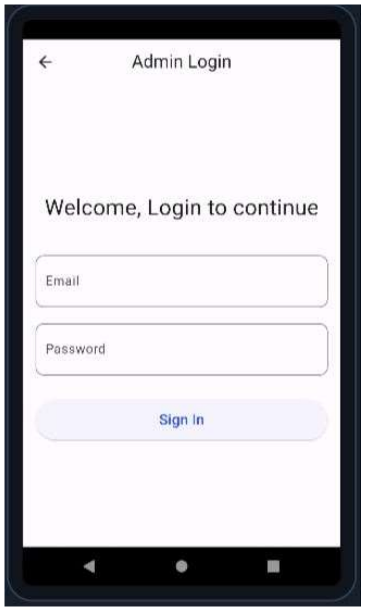
Service Provider Chat Page



Service Provider Profile Page



Admin Pages

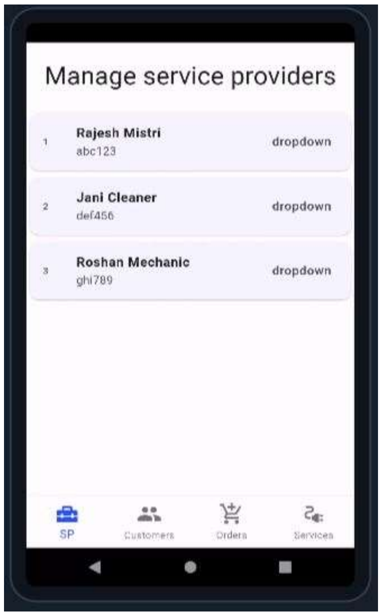

Admin Login

Welcome, Login to continue

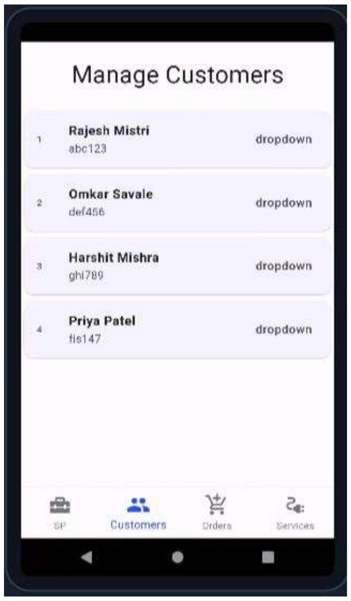
Email

Password

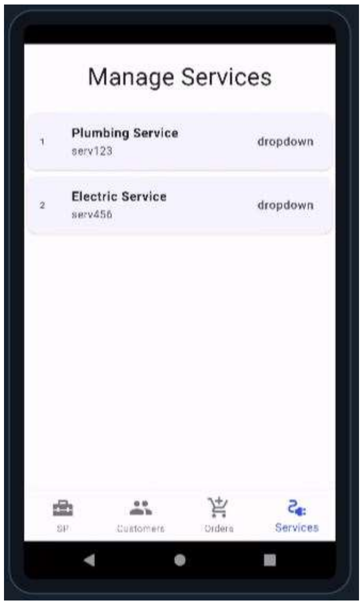
Sign In


Manage service providers

1	Rajesh Mistri	abc123	dropdown
2	Jani Cleaner	def456	dropdown
3	Roshan Mechanic	ghi789	dropdown

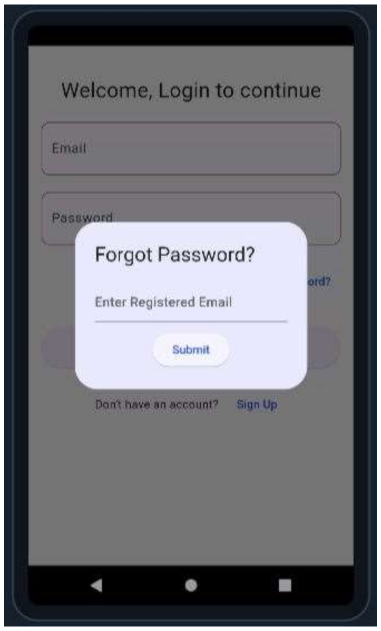

Manage Customers

1	Rajesh Mistri	abc123	dropdown
2	Omkar Savale	def456	dropdown
3	Harshit Mishra	ghi789	dropdown
4	Priya Patel	fis147	dropdown


Manage Services

1	Plumbing Service	serv123	dropdown
2	Electric Service	serv456	dropdown

Customer Pages


Welcome, Login to continue

Email

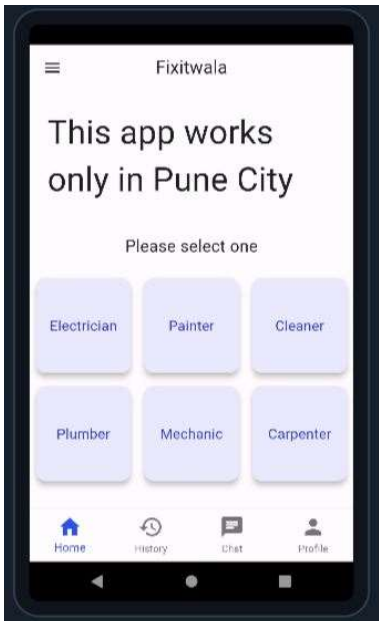
Password

Forgot Password?

Enter Registered Email

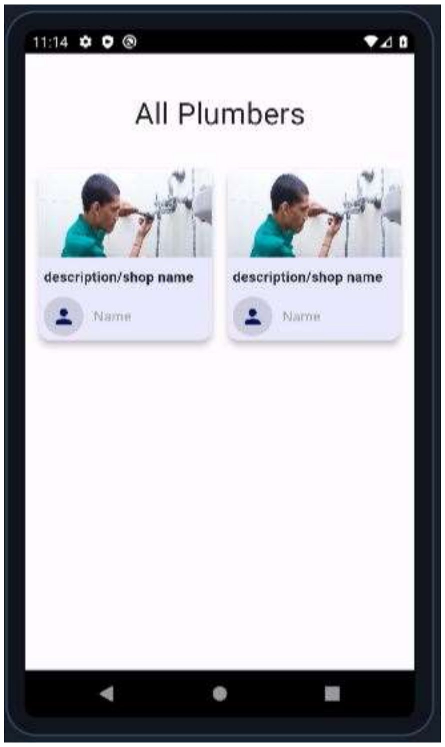
Submit

Don't have an account? Sign Up


Fixitwala

This app works only in Pune City

Please select one

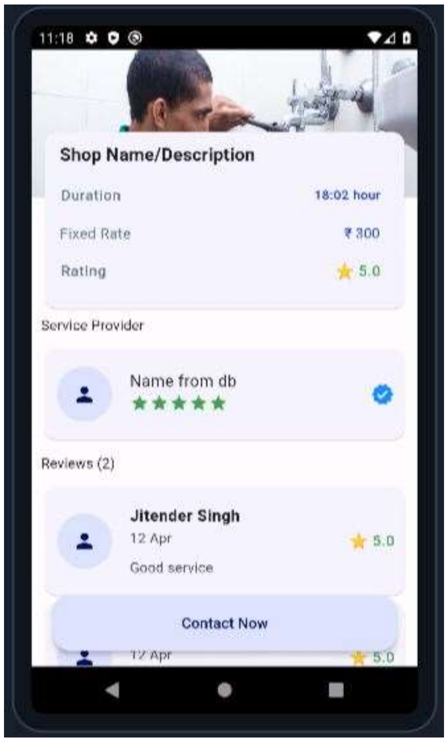

All Plumbers

description/shop name

Name

description/shop name

Name


11:18

Shop Name/Description

Duration 18:02 hour

Fixed Rate ₹ 300

Rating 5.0

Service Provider

Name from db

Reviews (2)

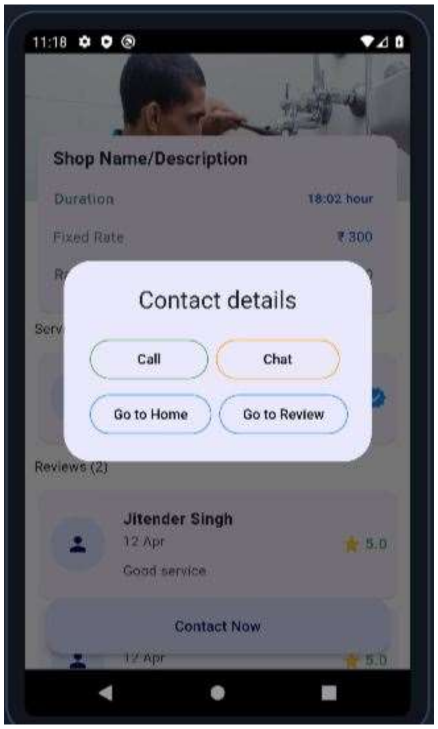
Jitender Singh

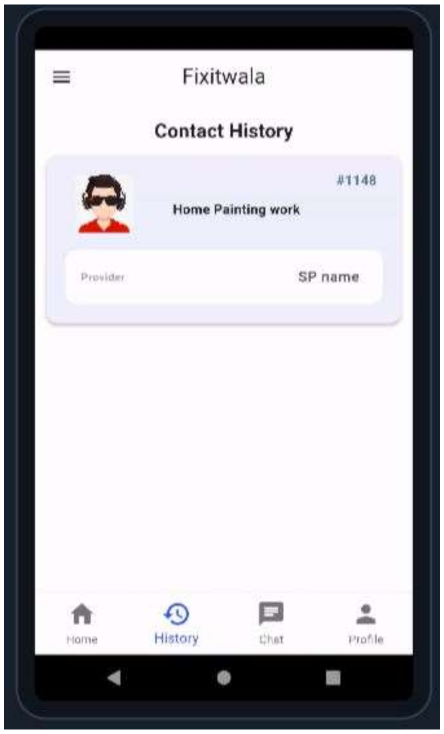
12 Apr

Good service

5.0

Contact Now



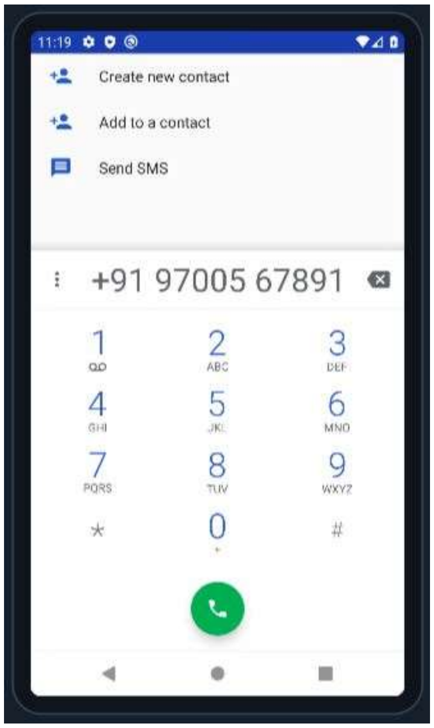

Fixitwala

Contact History

#1148

Home Painting work

Provider SP name



4.1 Algorithms

User Authentication and Authorization

1. When a user attempts to register:

1.1. Validate user input data:

- Ensure email format is valid.
- Check if password meets strength requirements.

1.2. Check if the email is not already registered:

- Query the database to see if the email exists.

1.3. Hash the password:

- Use a secure hashing algorithm to encrypt the password.

1.4. Store user data in the database:

- Insert user details (email, hashed password) into the user's table.

2. When a user attempts to log in:

2.1. Verify user credentials:

- Retrieve user data from the database based on the provided email.

- Compare the hashed password with the stored hash.

2.2. Generate and store a session token:

- If credentials are valid, create a unique session token.

- Store the session token in a session table or as a cookie.

3. Access control:

3.1. Restrict access based on user roles:

- Define user roles (e.g., seller, buyer).

- Associate permissions with each role.

3.2. Verify permissions:

- Check user role and permissions before allowing access to certain features.

Algorithm for chat page:

Initialization:

Initialize the chat page with the necessary UI elements such as message input field, send button, and message display area.

Load previous messages if any.

Display Messages:

Retrieve and display previous messages between the user and the service provider.

Messages should be displayed in chronological order, with timestamps to indicate when each message was sent.

Send Message:

When the user enters a message in the input field and clicks the send button:

Validate the message to ensure it is not empty.

Add the message to the message display area with the user's name and timestamp.

Send the message data to the backend server for storage and processing.

Receive Message:

Listen for incoming messages from the service provider or other users.

Upon receiving a new message:

Add the message to the message display area with the sender's name and timestamp.

Update the chat page to display the latest message at the bottom of the chat window.

Real-Time Updates:

Implement real-time updates using web sockets or a similar technology to ensure that messages are received and displayed instantly without requiring the user to refresh the page.

Error Handling:

Implement error handling to deal with cases such as failed message sending or receiving due to network issues or server errors.

Display appropriate error messages to the user and provide options for retrying or reporting the issue.

User Interaction:

Allow users to interact with messages by clicking on them to view details or reply.

Implement additional features such as emojis, file attachments, or message reactions based on project requirements.

Security:

Implement security measures such as end-to-end encryption to ensure that messages exchanged between users and service providers are secure and protected from unauthorized access.

Optimization:

Optimize the chat page algorithm and UI elements for performance, ensuring smooth and responsive user experience even with a large number of messages.

Service Provider Profile Page:

Initialization:

Load the profile details of the selected service provider.

Display Profile:

Display the service provider's name, contact information, ratings, reviews, services offered, and availability schedule.

Book Appointment:

Provide an option to book an appointment with the service provider.

Redirect the user to the appointment scheduling page with pre-filled service provider details.

Send Message:

Allow users to send messages to the service provider for inquiries or requests.

Redirect the user to the chat page with the service provider.

Service Listing Page:

Initialization:

Fetch and display a list of available services offered by different service providers.

Filter and Sort:

Provide options to filter services based on categories, location, ratings, and availability.

Allow users to sort services by relevance, ratings, or price.

Select Service:

Allow users to select a service to view details and choose a service provider.

View Details:

Display detailed information about the selected service, including descriptions, prices, and service provider profiles.

4.2 Coding

Main.dart

```
import 'package:myapp/customer/chat.dart';

import 'package:myapp/customer/contact_history.dart';

import 'package:myapp/customer/profile.dart';

import 'package:flutter/material.dart';

import 'package:myapp/landing_page.dart';

import 'package:myapp/my_drawer.dart';


void main() {

  runApp(const MyApp());

}


class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      title: 'Fixitwala',

      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.indigoAccent),

        useMaterial3: true,

      ),

      home: const MyHomePage(),

    );

  }

}


class MyHomePage extends StatefulWidget {

  const MyHomePage({super.key});

  @override

  State<MyHomePage> createState() => _MyHomePageState();

}


class _MyHomePageState extends State<MyHomePage> {

  int _selectedIndex = 0; // Track the selected index


  final List<Widget> _screens = const [

    LandingPage(),

    ContactHistory(),

    ChatPage(),

    ProfilePage(),
  ];


  @override

  Widget build(BuildContext context) {

    return SafeArea(

      child: Scaffold(

        appBar: AppBar(

          title: const Text('Fixitwala'),

          centerTitle: true,

        ),

        drawer: const MyDrawer(),

        body: _screens[_selectedIndex],

        bottomNavigationBar: BottomNavigationBar(

            iconSize: 30,

            items: const [

              BottomNavigationBarItem(

                icon: Icon(Icons.home),

                label: "Home",

              ),

              BottomNavigationBarItem(

                icon: Icon(Icons.history),

                label: "History",
              ),
              BottomNavigationBarItem(
                icon: Icon(Icons.chat_rounded),
                label: "Chat",
              ),
              BottomNavigationBarItem(
                icon: Icon(Icons.person),
                label: "Profile",
              ),
            ],
            currentIndex: _selectedIndex, // makes icon
active onTap
            onTap: (index) {
              setState(() {
                _selectedIndex = index;
              });
            },
            type: BottomNavigationBarType.fixed,
          ),
      ),
    );
  }
}
```

Landing_page.dart

```
import 'package:myapp/service/all_carpenter.dart';

import 'package:myapp/service/all_cleaner.dart';

import 'package:myapp/service/all_electrician.dart';

import 'package:myapp/service/all_mechanic.dart';

import 'package:myapp/service/all_painter.dart';

import 'package:myapp/service/all_plumber.dart';

import 'package:flutter/material.dart';


class LandingPage extends StatelessWidget {

  const LandingPage({super.key});


  final servicePages = const [

    ElectricianPage(),

    PainterPage(),

    CleanerPage(),

    PlumberPage(),

    MechanicPage(),

    CarpenterPage(),

  ];
  // Helper function to extract service name from page
type
  String getServiceName(Widget page) {
    return
page.runtimeType.toString().replaceAll(RegExp(r'Page$'),
'');
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(4.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            crossAxisAlignment:
CrossAxisAlignment.center,
            children: <Widget>[
              const Padding(
                padding: EdgeInsets.all(18.0),
                child: Text(
                  'This app works only in Pune City',
                  style: TextStyle(fontSize: 40),
                ),
              ),
              const SizedBox(height: 20),
              const Text(
                "Please select one",
                style: TextStyle(fontSize: 20),
              ),
              const SizedBox(height: 20),
              SizedBox(
                height: 350,
                child: GridView.builder(
                  gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
                    crossAxisCount: 3,
                    mainAxisSpacing: 8,
                    crossAxisSpacing: 8,
                  ),
                  itemCount: servicePages.length,
                  itemBuilder: (context, index) {
                    final servicePage =
servicePages[index];
```

```
                    final serviceName =
getServiceName(servicePage);
                    return _buildServiceCard(serviceName,
servicePage, context); // Pass both servicePage and
serviceName
                  },
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }


  Widget _buildServiceCard(String serviceName, Widget
servicePage, BuildContext context) {
    return InkWell(
      onTap: () => Navigator.push(
        context,
        MaterialPageRoute(builder: (context) =>
servicePage), // Use servicePage directly
      ),
      child: Card(
        elevation: 6,
        child: Center(
          child: Text(
            serviceName,
            style: const TextStyle(
              color: Color.fromARGB(255, 52, 49, 184),
              fontSize: 17,
            ),
          ),
        ),
      ),
    );
  }
}
```

My_drawer.dart

```
import 'package:flutter/material.dart';
import 'package:myapp/admin/admin_login.dart';
import 'package:myapp/customer/customer_login.dart';
import
'package:myapp/serviceProvider/service_provider_login.dar
t';
```

```
class MyDrawer extends StatelessWidget {
 const MyDrawer({super.key});

 @override
 Widget build(BuildContext context) => Drawer(
    child: Column(
     children: [
      Image.asset(
       'images/banner.jpg',
       fit: BoxFit.contain,
      ),
      Container(
       padding: const EdgeInsets.all(24),
       child: Wrap(
        runSpacing: 16,
        children: [
         ListTile(
          leading: const Icon(Icons.person),
          title: const Text('Customer'),
          onTap: () => Navigator.push(
           context,
                      MaterialPageRoute(
                        builder: (context) {
                          return const CustomerLogin();
                        },
                      ),
                    ),
                  ),
                  ListTile(
                    leading: const Icon(Icons.person_2),
                    title: const Text('Service
Provider'),
                    onTap: () => Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) {
                          return const SPLogin();
                        },
                      ),
                    ),
                  ),
                  ListTile(
                    leading: const Icon(Icons.person_3),
                    title: const Text('Admin'),
                    onTap: () => Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) {
                          return const AdminLogin();
                        },
                      ),
                    ),
                  )
                ],
              ),
            ),
          ],
        ),
      );
}
```

Admin_dashboard.dart

```
import 'package:myapp/admin/manage_customers.dart';
import 'package:myapp/admin/manage_orders.dart';
import 'package:myapp/admin/manage_services.dart';
```

```dart
import 'package:myapp/admin/manage_sp.dart';
import 'package:flutter/material.dart';

class AdminDashboard extends StatefulWidget {
  const AdminDashboard({super.key});

  @override
  State<AdminDashboard> createState() =>
_AdminDashboardState();
}

class _AdminDashboardState extends State<AdminDashboard>
{
  int _selectedIndex = 0; // Track the selected index

  final List<Widget> _screens = const [
    ManageSP(),
    ManageCustomers(),
    ManageOrders(),
    ManageServices(),
  ];

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        backgroundColor: Colors.grey,
        body: _screens[_selectedIndex],
        bottomNavigationBar: BottomNavigationBar(
          currentIndex: _selectedIndex, // makes icon
active onTap
          onTap: (index) {
            setState(() {
              _selectedIndex = index;
            });
          },
          iconSize: 30,
          items: const [
            BottomNavigationBarItem(
              icon: Icon(Icons.home_repair_service),
              label: "SP",
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.people),
              label: "Customers",
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.add_shopping_cart),
              label: "Orders",
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.electrical_services),
              label: "Services",
            ),
          ],
          type: BottomNavigationBarType.fixed,
        ),
      ),
    );
  }
}
```

Admin_login.dart

```
import 'package:myapp/admin/admin_dashboard.dart';
import 'package:flutter/material.dart';

class AdminLogin extends StatefulWidget {
  const AdminLogin({super.key});
  @override
  State<AdminLogin> createState() => _AdminLoginState();
}

class _AdminLoginState extends State<AdminLogin> {
  final GlobalKey<FormState> _formKey =
GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _pwdController = TextEditingController();
  String email = '';
  String pwd = '';

  String? _validateEmail(value) {
    if (value!.isEmpty) {
      return 'Please enter email';
    }
    RegExp emailRegExp = RegExp(r'^[\w-\.]+@([\w-
]+\.)+[\w-]{2,4}$');
    if (!emailRegExp.hasMatch(value)) {
      return 'Please enter a valid email';
    }
    return null;
  }


  String? _validatePassword(value) {

    if (value!.isEmpty) {

      return 'Please enter password';

    }

    if (value.length < 8) {

      return 'Please enter password with minimum 8
characters';

    }

    return null;

  }


  void _submitform() {

    email = _emailController.text;

    pwd = _pwdController.text;

    if (_formKey.currentState!.validate()) {

      if (email == "admin-rahul@gmail.com" && pwd ==
"12345678") {

        ScaffoldMessenger.of(_formKey.currentContext!)

            .showSnackBar(const SnackBar(content:
Text("Login successful")));

        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => const
AdminDashboard()),
        );
      } else {
        ScaffoldMessenger.of(_formKey.currentContext!)
            .showSnackBar(const SnackBar(content:
Text("Invalid credentials")));
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        appBar: AppBar(
          title: const Text('Admin Login'),
          centerTitle: true,
        ),
        body: Center(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: SingleChildScrollView(
              child: Form(
                key: _formKey,
                child: Column(
                  children: [
                    const Text(
                      "Welcome, Login to continue",
                      style: TextStyle(
                        fontSize: 27,
                      ),
                    ),
                    const SizedBox(height: 40),
                    TextFormField(
                      controller: _emailController,
                      decoration: InputDecoration(
                        labelText: 'Email',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      keyboardType:
TextInputType.emailAddress,
                      validator: _validateEmail,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 20),
                    TextFormField(
                      controller: _pwdController,
                      obscureText: true,
                      decoration: InputDecoration(
                        labelText: 'Password',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      validator: _validatePassword,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 30),
                    SizedBox(
                      height: 50,
                      width: double.infinity,
                      child: ElevatedButton(
                        onPressed: _submitform,
                        child: const Text(
                          "Sign In",
                          style: TextStyle(fontSize: 18),
                        ),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Manage_customers.dart

```
import 'package:flutter/material.dart';
```

```
// TODO: implement using db

class Customer {
  final int sno;
  final String customerName;
  final String customerId;

  const Customer({
    required this.sno,
    required this.customerName,
    required this.customerId,
  });
}

class ManageCustomers extends StatelessWidget {
  const ManageCustomers({super.key});

  final List<Customer> customers = const [
    Customer(sno: 1, customerName: "Rajesh Mistri",
customerId: "abc123"),
    Customer(sno: 2, customerName: "Omkar Savale",
customerId: "def456"),
    Customer(sno: 3, customerName: "Harshit Mishra",
customerId: "ghi789"),
    Customer(sno: 4, customerName: "Priya Patel",
customerId: "fis147"),
  ];


  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        body: Column(
          children: [
            const Padding(
              padding: EdgeInsets.symmetric(horizontal:
5, vertical: 20),
              child: Text(
                "Manage Customers",
                style: TextStyle(
                  fontSize: 30,
                ),
              ),
            ),
            Expanded(
       // Use Expanded to allow scrolling for the
list
       child: ListView.builder(
        itemCount: customers.length, // Number of
cards to create
        itemBuilder: (context, index) {
         final customer = customers[index];
         return Card(
          child: ListTile(
           leading: Text("${customer.sno}"),
           title: Text(
            customer.customerName,
            style: const
TextStyle(fontWeight: FontWeight.bold),
           ),
           subtitle:
Text(customer.customerId),
           trailing: const Text(
            "dropdown",
            style: TextStyle(fontSize: 15),
           ),
          ),
         );
        },
              ),

            ),

          ],

        ),

      ),

    );

  }

}
```

Manage_orders.dart

```
import 'package:flutter/material.dart';


// TODO: use a table instead of card for orders

// TODO: implement using db


class Order {

  final int sno;

  final String spName;

  final String orderId;


  const Order({
    required this.sno,

    required this.spName,

    required this.orderId,

  });

}


class ManageOrders extends StatelessWidget {

  const ManageOrders({super.key});


  final List<Order> orders = const [

    Order(sno: 1, spName: "Lakhan Painter", orderId:
"#123"),

    Order(sno: 2, spName: "Lalu Electrician", orderId:
"#456"),

  ];


  @override

  Widget build(BuildContext context) {

    return SafeArea(

      child: Scaffold(

        body: Column(

          children: [

            const Padding(
              padding: EdgeInsets.symmetric(horizontal:
5, vertical: 20),
              child: Text(
                "Manage Orders",
                style: TextStyle(
                  fontSize: 30,
                ),
              ),
            ),
            Expanded(
              // Use Expanded to allow scrolling for the
list
              child: ListView.builder(
                itemCount: orders.length, // Number of
cards to create
                itemBuilder: (context, index) {
                  final order = orders[index];
                  return Card(
                    child: ListTile(
                      leading:
Text(order.sno.toString()),
                      title: Text(order.spName),
                      subtitle: Text(order.orderId),
                      trailing: const Text("Use Table
here"),
                    ),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Chat.dart

```
import 'package:flutter/material.dart';

class IndividualChat extends StatefulWidget {
  const IndividualChat({super.key});

  @override
  State<IndividualChat> createState() =>
_IndividualChatState();

}


class _IndividualChatState extends State<IndividualChat>
{

  final _textController = TextEditingController(); //
Create the controller

  List<String> messages = ["Hi"];


  void sendMessage(String message) {

    setState(() {

      messages.add(message);

    });

  }


  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Rajesh"),

      ),

      body: Stack(
        children: [
          Positioned.fill(
            child: Container(
              color: Colors.blueGrey,
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              mainAxisSize: MainAxisSize.min, // Restrict
column size
              crossAxisAlignment: CrossAxisAlignment.end,
// Align to right
              children: [
                // Display the list of messages
                Expanded(
                  child: ListView.builder(
                    itemCount: messages.length,
                    itemBuilder: (context, index) =>
Column(
                      crossAxisAlignment:
CrossAxisAlignment.end,
                      children: [
                        Card(
                          elevation: 0,
                          child: Padding(
                            padding: const
EdgeInsets.all(15.0),
                            child: Text(messages[index]),
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
                const SizedBox(height: 10.0),
                Row(
                  children: [
                    Expanded(
                      child: TextField(
                        controller: _textController,
                        decoration: InputDecoration(
                          hintText: "Message",
                          hintStyle: const
TextStyle(color: Colors.grey),
                          fillColor: Colors.white,
                          filled: true,
                          border: OutlineInputBorder(
                            borderRadius:
BorderRadius.circular(30.0),
                          ),
                        ),
                      ),
                    ),
                    const SizedBox(width: 10.0),
                    Container(
                      decoration: const BoxDecoration(
                        shape: BoxShape.circle,
                        color: Colors.white,
                      ),
                      child: IconButton(
                        padding: const
EdgeInsets.all(15),
                        onPressed: () {
                          final message =
_textController.text;
                          if (message.isNotEmpty) {
                            sendMessage(message);
                            _textController.clear();
                          }
                        },
                        icon: const Icon(
                          Icons.send,
                          size: 30,
                          color: Colors.black,
                        ),
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

Customer_login.dart

```
import 'package:myapp/customer/profile.dart';
import 'package:flutter/material.dart';

// import
'package:fixitwala/customer/customer_dashboard.dart';

import 'package:myapp/customer/customer_register.dart';


class CustomerLogin extends StatefulWidget {

  const CustomerLogin({super.key});


  @override

  State<CustomerLogin> createState() =>
_CustomerLoginState();

}


class _CustomerLoginState extends State<CustomerLogin> {

  final GlobalKey<FormState> _formKey =
GlobalKey<FormState>();

  final _alertformKey = GlobalKey<FormState>();

  String _email = "";

  String _forgotPasswordStatus = '';

  final _emailRegex =

      RegExp(r'^[\w-]+(\.[\w-]+)*@([a-zA-Z0-9-]+)(\.[a-
zA-Z]{2,})+$');


  void _submitform() {
    if (_formKey.currentState!.validate()) {

ScaffoldMessenger.of(_formKey.currentContext!).showSnackB
ar(
        const SnackBar(
          content: Text("Login successful"),
        ),
      );
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) {
            return const ProfilePage();
            // return const CustomerDashboard();
          },
        ),
      );
    }
  }

  String? _validateEmail(value) {
    if (value!.isEmpty) {
      return 'Please enter email';
    }

    RegExp emailRegExp = RegExp(r'^[\w-\.]+@([\w-
]+\.)+[\w-]{2,4}$');

    if (!emailRegExp.hasMatch(value)) {

      return 'Please enter a valid email';

    }

    return null;

  }


  String? _validatePassword(value) {

    if (value!.isEmpty) {

      return 'Please enter password';

    }

    if (value.length < 8) {

      return 'Please enter password with minimum 8
characters';

    }

    return null;

  }


  void _forgotPassword() {

    showDialog(

      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Forgot Password?'),
          content: Form(
            key: _alertformKey,
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                TextFormField(
                  keyboardType:
TextInputType.emailAddress,
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Please enter your email
address';
                    } else if
(!_emailRegex.hasMatch(value)) {
                      return 'Please enter a valid email
address';
                    }
                    return null;
                  },
                  onSaved: (newValue) => _email =
newValue!,
                  decoration: const InputDecoration(
                    hintText: 'Enter Registered Email',
                  ),
                ),
                const SizedBox(height: 10),
                ElevatedButton(
                  onPressed: () {
                    if
(_alertformKey.currentState!.validate()) {
                      _alertformKey.currentState!.save();
                      // Replace with your actual logic
to send reset link via email
                      // print('Sending reset link to
$_email');


                      // TODO: check db for registered
email
                      setState(() {
                        _forgotPasswordStatus =
                            'Password sent to registered
email: $_email';
                      });
                      Navigator.of(context).pop();
                    }
                  },
                  child: const Text('Submit'),
                ),
              ],
            ),
          ),
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.indigoAccent),
        useMaterial3: true,
      ),
      home: SafeArea(
        child: Scaffold(
          body: Padding(
            padding: const EdgeInsets.all(16.0),
            child: SingleChildScrollView(
              child: Form(
                key: _formKey,
                child: Column(
                  children: [
                    const SizedBox(height: 20),
                    const Text(
                      "Welcome, Login to continue",
                      style: TextStyle(
                        fontSize: 25,
                      ),
                    ),
                    const SizedBox(height: 20),
                    TextFormField(
                      decoration: InputDecoration(
                        labelText: 'Email',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      keyboardType:
TextInputType.emailAddress,
                      validator: _validateEmail,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 20),
                    TextFormField(
                      obscureText: true,
                      decoration: InputDecoration(
                        labelText: 'Password',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      validator: _validatePassword,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 20),
                    Row(
                      mainAxisAlignment:
MainAxisAlignment.end,
                      children: [
                        TextButton(
                          onPressed: _forgotPassword,
                          child: const Text('Forgot
Password?'),
                        ),
                      ],
                    ),
                    const SizedBox(height: 30),
                    SizedBox(
                      height: 50,
                      width: double.infinity,
                      child: ElevatedButton(
                        onPressed: _submitform,
                        child: const Text(
                          "Sign In",
                          style: TextStyle(fontSize: 18),
                        ),
                      ),
                    ),
                    const SizedBox(height: 20),
                    Row(
                      mainAxisAlignment:
MainAxisAlignment.center,
                      children: [
                        const Text("Don't have an
account?"),
                        const SizedBox(width: 10),
                        TextButton(
                          onPressed: () {
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                builder: (context) {
                                  return const
CustomerRegister();
                                },
                              ),
                            );
                          },
                          child: const Text("Sign Up"),
                        ),
                      ],
                    ),
                    const SizedBox(height: 20),
                    Text(

                      _forgotPasswordStatus,

                      style: const TextStyle(color:
Colors.green),

                    ),

                  ],

                ),

              ),

            ),

          ),

        ),

      ),

    );

  }

}
```

Customer_registration.dart

```
import 'package:myapp/customer/profile.dart';

import 'package:flutter/material.dart';

import 'package:myapp/customer/customer_login.dart';
```

```
class CustomerRegister extends StatefulWidget {
  const CustomerRegister({super.key});


  @override
  State<CustomerRegister> createState() =>
_CustomerRegisterState();
}


class _CustomerRegisterState extends
State<CustomerRegister> {
  final GlobalKey<FormState> _formKey =
GlobalKey<FormState>();


  void _submitform() {
    if (_formKey.currentState!.validate()) {

ScaffoldMessenger.of(_formKey.currentContext!).showSnackB
ar(
        const SnackBar(
          content: Text("Registration successful"),
        ),
      );
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) {
            return const ProfilePage();
            // return const CustomerDashboard();
          },
        ),
      );
    }
  }

  String? _validateName(value) {
    if (value!.isEmpty) {
      return 'Please enter Name';
    }
    RegExp nameRegExp = RegExp(r'^([a-zA-Z]+\s)*[a-zA-
Z]+$');
    if (!nameRegExp.hasMatch(value)) {
      return 'Please enter a valid name';
    }
    if (value.length < 3) {
      return 'Please enter full name';
    }
    return null;
  }


  String? _validateAddress(value) {

    if (value!.isEmpty) {

      return 'Please enter Address';

    }

    if (value.length < 10) {

      return 'Please enter valid address';

    }

    return null;

  }


  String? _validateEmail(value) {

    if (value!.isEmpty) {

      return 'Please enter email';

    }

    RegExp emailRegExp = RegExp(r'^[\w-\.]+@([\w-
]+\.)+[\w-]{2,4}$');

    if (!emailRegExp.hasMatch(value)) {

      return 'Please enter a valid email';

    }

    return null;

  }
  String? _validateMobile(value) {
    if (value!.isEmpty) {
      return 'Please enter phone number';
    }
    RegExp mobileRegExp = RegExp(r'^[6789]\d{9}$');
    if (!mobileRegExp.hasMatch(value)) {
      return 'Please enter a valid Number';
    }
    return null;
  }

  String? _validatePassword(value) {
    if (value!.isEmpty) {
      return 'Please enter password';
    }
    if (value.length < 8) {
      return 'Please enter password with minimum 8
characters';
    }
    return null;
  }
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.indigoAccent),
        useMaterial3: true,
      ),
      debugShowCheckedModeBanner: false,
      home: SafeArea(
        child: Scaffold(
          body: Padding(
            padding: const EdgeInsets.all(16.0),
            child: SingleChildScrollView(
              child: Form(
                key: _formKey,
                child: Column(
                  children: [
                    const SizedBox(height: 20),
                    const CircleAvatar(
                      radius: 40,
                      child: Icon(Icons.person, size:
60),
                    ),
                    const SizedBox(height: 20),



                    const Text(
                      "Hello User !",
                      style: TextStyle(
                        fontSize: 25,
                      ),
                    ),
                    const SizedBox(height: 20),



                    const Text("Signup For Better
Experience"),
                    const SizedBox(height: 20),



                    TextFormField(
                      decoration: InputDecoration(
                        labelText: 'Full Name',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      validator: _validateName,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 20),


                    TextFormField(
                      decoration: InputDecoration(
                        labelText: 'Address',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      validator: _validateAddress,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(
                      height: 20,
                    ),



                    TextFormField(
                      decoration: InputDecoration(
                        labelText: 'Email',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      keyboardType:
TextInputType.emailAddress,
                      validator: _validateEmail,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(
                      height: 20,
                    ),
                    TextFormField(
                      decoration: InputDecoration(
                        labelText: 'Contact Number',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      keyboardType: TextInputType.phone,
                      validator: _validateMobile,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 20),


                    TextFormField(
                      obscureText: true,
                      decoration: InputDecoration(
                        labelText: 'Password',
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(10),
                        ),
                      ),
                      validator: _validatePassword,
                      autovalidateMode:
AutovalidateMode.onUserInteraction,
                    ),
                    const SizedBox(height: 30),
                    SizedBox(
                      height: 50,
                      width: double.infinity,
                      child: ElevatedButton(
                        onPressed: _submitform,
                        child: const Text(
                          "Sign Up",
                          style: TextStyle(fontSize: 18),
                        ),
                      ),
                    ),
                    const SizedBox(height: 15),
                    Row(
                      mainAxisAlignment:
MainAxisAlignment.center,
                      children: [
                        const Text("Already have an
account?"),
                        const SizedBox(width: 10),
                        TextButton(
                          onPressed: () {
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                builder: (context) {
                                  return const
CustomerLogin();
                                },
                              ),
                            );
                          },
                          child: const Text("Sign In"),
                        ),
                      ],
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

5.1 Test Strategy

A comprehensive test strategy for the "Fixitwala: home services" app platform would ensure that the system meets its functional and non-functional requirements, is reliable, secure, and delivers a seamless user experience.

Here's an outline of the test strategy:

1. Test Planning:

- Define test objectives, scope, and timelines.
- Identify key features and functionalities to be tested.
- Allocate resources, including testing team members and testing environments.
- Develop a test plan outlining test scenario, test cases, and testing techniques.

2. Test Environment Setup:

- Set up testing environments that replicate production conditions, including hardware, software, and network configurations.

- Ensure compatibility with various web browsers, operating systems, and devices.

- Implement tools for test management, defect tracking, and performance monitoring.

3. Functional Testing:

- Verify that all functional requirements are implemented correctly and meet user expectations.

- Conduct test scenarios covering user registration, property listings, property viewing, and user profiles.

- Validate data input validation, error handling, and edge cases.

4. Usability Testing:

- Evaluate the user interface design, navigation, and overall usability of the platform.

- Gather feedback from representative users to identify usability issues and areas for improvement.

- Ensure accessibility for users with disabilities, adhering to accessibility standards (WCAG).

5. Performance Testing:

- Measure system performance under normal and peak load conditions.

- Conduct load testing, stress testing, and scalability testing to identify performance bottlenecks and optimize system resources.

- Monitor response times, throughput, and resource utilization to ensure acceptable performance levels.

6. Security Testing:

- Identify and address potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), and authentication flaws.

- Perform penetration testing to assess the resilience of the platform against external attacks.

- Implement security measures such as encryption, secure authentication mechanisms, and data protection controls.

7. Compatibility Testing:

- Validate platform compatibility across different web browsers (Chrome, Firefox, Edge, Safari), operating systems (Windows, Linux, macOS, Android, iOS), and devices (desktops, laptops, smartphones, tablets).

- Ensure responsiveness and consistent user experience across various screen sizes and resolutions.

8. Regression Testing:

- Conduct regression testing to verify that new updates or fixes do not introduce regression defects.

- Re-run previously executed test cases to ensure that existing functionalities remain intact after changes are made to the system.

9. Documentation and Reporting:

- Document test plans, test cases, test results, and any defects found during testing.

- Generate test reports summarizing testing activities, findings, and recommendations for improvement.

- Communicate test results to stakeholders and collaborate on addressing identified issues.

By following this test strategy, the "Fixitwala: home services" app platform can undergo thorough testing to ensure quality, reliability, security, and usability before being deployed to production and made available to brokers and buyers in the Pune real estate market.

5.2 Unit Test Plan

1. Identify Units

Identify the individual components and modules of the home services app that will be tested. This could include components such as user authentication, service search, messaging system, payment gateway, appointment scheduling, etc.

2. Define Test Cases

Define test cases for each unit that specify the inputs, expected outputs, and any other conditions or criteria for the test. For example:

User Authentication:

Test Case 1: Verify that valid credentials allow users to log in successfully.

Test Case 2: Verify that invalid credentials result in appropriate error messages.

Test Case 3: Test the "Forgot Password" functionality and ensure that users can reset their passwords successfully.

Service Search:

Test Case 1: Test the search functionality with valid search queries and ensure that relevant service providers are returned.

Test Case 2: Test the search functionality with invalid search queries and ensure appropriate error handling.

Test Case 3: Test the application of filters and sorting options to search results.

Messaging System:

Test Case 1: Test sending messages between users and service providers and verify successful delivery.

Test Case 2: Test sending messages with attachments (e.g., images) and verify successful transmission.

Test Case 3: Test handling of message notifications and ensure users receive timely notifications.

3. Create Test Code

Write test code using the appropriate testing framework (e.g., pytest, unittest) to automate the execution of the test cases. This code will simulate the inputs and verify the outputs of each unit.

4. Execute Tests

Execute the test code to run the unit tests. The tests should be run regularly during the development process to catch and fix issues early.

5. Review and Refine

Review the test results and refine the test cases as needed. If any issues are identified, they should be fixed, and the tests rerun to ensure that the fixes are effective.

6. Document Results

Document the results of the unit tests, including any issues found and their resolutions. This documentation will serve as a record of the testing process and help track the progress of the project.

By following this unit test plan, we aim to ensure that each component of the home services app functions correctly on its own, laying a solid foundation for the integration and testing of the system as a whole.

5.3 Acceptance Test Plan

The acceptance test plan is a critical phase of testing that validates whether the system meets the specified requirements and aligns with user expectations. This process involves creating test cases based on user stories and acceptance criteria, which are essentially detailed descriptions of how a new feature should function from the perspective of the end user.

During acceptance testing, stakeholders and end users are involved in testing the system to ensure that it meets their needs and functions as intended. This phase helps ensure that the system is ready for deployment and meets the business goals set out for it.

Key aspects of the acceptance test plan include:

Test Case Creation: Test cases are created based on user stories and acceptance criteria. These test cases outline the steps to be performed, the expected results, and the actual results observed during testing.

User Involvement: Stakeholders and end users are actively involved in the testing process. Their feedback and validation are crucial in determining if the system meets their requirements and expectations.

Functional and Usability Testing: The focus is on both functional testing (ensuring the system functions as expected) and usability testing (ensuring the system is easy to use and intuitive).

Validation of Requirements: Each requirement is validated to ensure that it has been implemented correctly and meets the user's needs.

Bug Reporting and Resolution: Any issues or bugs identified during acceptance testing are reported, prioritized, and resolved before deployment.

Documentation: The acceptance test plan should be well-documented, detailing the test cases, test results, and any issues encountered during testing.

Overall, the acceptance test plan plays a crucial role in ensuring that the system meets user expectations and is ready for deployment. It helps validate that the system functions as intended and provides a positive user experience.

5.4 Test Case / Test Script

Users:

TEST CASE ID	SCENARIO TO TEST	STEPS TO PERFORMANCE	EXPECTED RESULTS	ACTUAL RESULTS	PASS/FAIL
TC1	Sign Up	1. Enter name, email address, password, confirm password 2. Click on the sign up button.	App should expect valid details enter by user and should redirect user to their home page.	Registration into Application is successful.	Pass
TC2	Login	1. Clicks on the login link. 2. Enter the valid username and password. 3. Click on the login in button.	App should expect valid details entered by the user and should redirect user to home page.	Login to Application is successful.	Pass

TC2.1	Login	1. Open the Login page. 2. Enter invalid username 3. Click on the sign in button.	Website should not accept invalid email address. website should throw message. “Invalid Credentials”.	Login denied with appropriate message.	Pass

TC2.2	Login	1. Open the Login page. 2.Enter the valid username. 3. Enter invalid password. 4. Click on the login button.	Website should not accept invalid password. Website should throw message "Invalid Credentials"	Login denied with appropriate message.	Pass
TC3	Search Service	Search the Searvice with their name on Application.	Application show the service.	Can easily access that service.	Pass
TC3.1	Profile	Customers can upload profile picture, add names and other details.	Customer should be able to upload pic, add names and details	User's profile should display the uploaded picture and all details.	Pass

TC4	Chat	Select a user/service provider, Compose message and Click "Send"	User should be able to send message.	Message is sent successfully	Pass
TC5	Call Functionality	User should be able to call the service provider on click of call button.	User should be able to call the service provider.	Call rings and received / rejected by the service provider	Pass

5.5 Defect report/ Test Log

Defect Report:

Defect ID	Module	Description	Severity	Status
DEF001	User Authentication	Invalid credentials do not show error message	High	Open
DEF002	Service Search	Filters not applied correctly	Medium	Closed
DEF003	Messaging System	Messages not delivered in real-time	High	Open
DEF004	Payment Gateway	Payment confirmation email not sent	Low	Open
DEF005	Call Functionality	Call button not visible on service provider profile	Medium	Closed

Test Summary:

Test Case ID	Description	Status
TC_AUTH_001	Valid User Login	Passed
TC_AUTH_002	Invalid User Login	Passed
TC_SEARCH_001	Valid Service Search	Passed
TC_SEARCH_002	Empty Search	Passed
TC_MSG_001	Send Message	Passed
TC_MSG_002	Receive Message	Passed
TC_PAY_001	Successful Payment	Failed
TC_PAY_002	Payment Error	Passed

The limitations of the proposed home services app could include:

Scalability: While the system is designed to accommodate a growing user base, there may be scalability limitations as the number of users and content grows. This could lead to performance issues and slower response times.

Security: Despite efforts to enhance user privacy and security, there may still be vulnerabilities that could be exploited by malicious users or hackers. Continuous monitoring and updates are necessary to mitigate security risks.

User Experience: While efforts have been made to create a user-friendly interface, there may still be aspects of the system that are confusing or difficult for users to navigate. User feedback will be crucial for improving the user experience over time.

Dependency on Third-Party Services: The system may rely on third-party services for certain functionalities, such as image and video hosting or payment processing. Any issues with these services could impact the overall functionality of the system.

Data Privacy and Compliance: Despite efforts to protect user data, there may still be challenges in ensuring compliance with data privacy regulations and protecting user information from unauthorized access or misuse.

Technical Support and Maintenance: The system will require ongoing technical support and maintenance to address any issues that arise and to keep the system up-to-date with the latest technologies and security patches.

Cost: Developing and maintaining the system may require significant financial resources, including costs associated with hosting, infrastructure, and personnel.

Overall, while the proposed system aims to address the needs of users and provide a platform for home services, there are limitations and challenges that need to be considered and addressed to ensure the success of the system.

Proposed enhancements for the home services app project could include:

Real-Time Tracking: Integrate real-time tracking features so users can track the location of service providers en route to their location. This enhances transparency and reduces uncertainty about service arrival times.

Secure Payment Gateway: Ensure the app has a secure payment gateway that supports multiple payment methods, including credit/debit cards, digital wallets, and possibly even cryptocurrencies, to facilitate seamless transactions.

Multi-Language Support: Cater to a wider user base by implementing multi-language support within the app. This can enhance accessibility and user engagement, especially in regions with diverse linguistic backgrounds.

Appointment Scheduling System: Develop a robust appointment scheduling system that allows users to book services at their preferred date and time. Integration with calendar apps and reminders can further improve user experience.

AI-Powered Recommendations: Utilize artificial intelligence algorithms to analyze user preferences and behavior, providing personalized recommendations for service providers or related services based on past interactions.

Emergency Assistance Feature: Introduce an emergency assistance feature where users can quickly request urgent services such as plumbing or locksmith services. Implementing a dedicated hotline or emergency button can ensure timely assistance during critical situations.

Social Media Integration: Allow users to share their experiences or recommend service providers through social media integration. This can help in expanding the app's reach and attracting new users through word-of-mouth referrals.

In conclusion, the development of the Fixitwala project has been a rewarding experience, presenting challenges that have been met with innovative solutions. By leveraging technologies such as Flutter, Windows, and SQLite, I have created a platform that aims to revolutionize home services online.

The project's objectives of providing a user-friendly interface, enhancing user privacy and security, implementing a scalable architecture, and offering personalized user experiences have been successfully achieved. The system caters to a wide range of users, including individuals, professionals, and communities, providing a versatile and engaging platform for home services.